UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	ISSUE DATE	PATENT NO.	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/408,681	01/06/2015	8930538	0137-US-U1	9965

83579 7590 12/17/2014

LEVEL 3 COMMUNICATIONS, LLC
Attn: Patent Docketing
1025 Eldorado Blvd.
Broomfield, CO 80021

ISSUE NOTIFICATION

The projected patent number and issue date are specified above.

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(application filed on or after May 29, 2000)

The Patent Term Adjustment is 645 day(s). Any patent to issue from the above-identified application will include an indication of the adjustment on the front page.

If a Continued Prosecution Application (CPA) was filed in the above-identified application, the filing date that determines Patent Term Adjustment is the filing date of the most recent CPA.

Applicant will be able to obtain more detailed information by accessing the Patent Application Information Retrieval (PAIR) WEB site (http://pair.uspto.gov).

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Application Assistance Unit (AAU) of the Office of Data Management (ODM) at (571)-272-4200.

APPLICANT(s) (Please see PAIR WEB site http://pair.uspto.gov for additional applicants):

David Fullagar, Boulder, CO;
Christopher Newton, Thousand Oaks, CA;
Laurence R. Lipstone, Calabasas, CA;

The United States represents the largest, most dynamic marketplace in the world and is an unparalleled location for business investment, innovation, and commercialization of new technologies. The USA offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to encourage and facilitate business investment. To learn more about why the USA is the best country in the world to develop technology, manufacture products, and grow your business, visit SelectUSA.gov.

0041

Change(s) applied to document, /Q.N./ 11/12/2014

Please replace **paragraph ~~0043~~** with the following:

[0041] If the object is not available on the edge server **108** or on a peer, the edge server **108** ascertains whether this object is served based on its popularity (i.e., whether this object has been designated so that the object's popularity will be used to determine where it will be served from) (at **506**). If so, then the request is sent to the popularity service **102** associated with the edge server **108**, in this case, to the popularity server for ~~the~~ this group (**S3a**).

Please replace **paragraph 0043** with the following:

[0043] It is preferably to allow for a mix of edge servers, some performing popularity checking (as described above), while others do not. For those that are not running the popularity service, the name (hostname) used to request an object will resolve to a parent server (that may or may not provide popularity services). If the parent server does not provide popularity services, then the content will be obtained by the edge server from that parent server, and the content will be served to the client. On the other hand, if that parent server does provide popularity services, it can determine whether or not the edge server is a Longtail coserver based, e.g., on the IP (Internet Protocol) address of the edge server. For non-coservers, the parent server can handle the request without any popularity processing.

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	6	of	21	Attorney Docket No: 0137-US-U1

Change(s) applied to document, /R.K.C./ 12/5/2014

US PATENT DOCUMENTS					
Examiner Initial *	**Cite No**	**Document Number**	**Publication Date**	**Name of Patentee or Applicant of Cited Document**	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5913028	Jun 15, 1999	Wang, F. et al.	
		5913033	Jun 15, 1999	Grout, G. B.	
		5918010	Jun 29, 1999	Appleman, K. H., et al.	
		5919247	Jul 6, 1999	~~Van, H. A., et al.~~	Van Hoff et al.
		5920701	Jul 6, 1999	Miller, C. K., et al.	
		5931904	Aug 3, 1999	Banga, Gaurav	
		5933832	Aug 3, 1999	Suzuoka, T. et al.	
		5933835	Aug 3, 1999	Adams, Robert et al.	
		5935207	Aug 10, 1999	Logue, J. D., et al.	
		5940831	Aug 17, 1999	Takano, H.	
		5944780	Aug 31, 1999	Chase, J. S.	
		5951694	Sep 14, 1999	Choquier, P.	
		5956489	Sep 21, 1999	San Andres, R. J., et al.	
		5956716	Sep 21, 1999	Kenner, B. et al.	
		5958008	Sep 28, 1999	Pogrebisky, M. et al.	
		5961596	Oct 5, 1999	Takubo, S. et al.	
		5966440	Oct 12, 1999	Hair, A. R.	
		5968121	Oct 19, 1999	Logan, K. W., et al.	
		5973696	Oct 26, 1999	Agranat, I. D., et al.	
		5978791	Nov 2, 1999	Farber, D. A., et al.	
		5983214	Nov 9, 1999	Lang, A. K., et al.	
		5983227	Nov 9, 1999	Nazem, F. et al.	
		5987606	Nov 16, 1999	Cirasole, P. et al.	
		5991809	Nov 23, 1999	Kriegsman, M. E., et al.	
		5996025	Nov 30, 1999	Day	
		6002720	Dec 14, 1999	Yurt, P. et al.	
		6003030	Dec 14, 1999	Kenner, B. et al.	
		6012090	Jan 4, 2000	Chung, Pi-Yu. et al.	
		6014686	Jan 11, 2000	Elnozahy, E. N., et al.	
		6014698	Jan 11, 2000	Griffiths, M. J.	
		6016509	Jan 18, 2000	Dedrick, Rick	
		6016512	Jan 18, 2000	Huitema, Christian	
		6018516	Jan 25, 2000	Packer, R. L.	
		6021426	Feb 1, 2000	Douglis, F.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	5	of	21	Attorney Docket No: 0137-US-U1

Change(s) applied to document, /R.K.C./ 12/5/2014

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5774668	Jun 30, 1998	Choquier, P. et al.	
		5777989	Jul 7, 1998	McGarvey, J.	
		5778187	Jul 7, 1998	Monteiro, Antonio M., et al.	
		5781909	~~Jul 14, 2009~~	Logan, James et al.	7/1998
		5784058	Jul 21, 1998	LaStrange, T. E., et al.	
		5796952	Aug 18, 1998	Davis, O. et al.	
		5799141	Aug 25, 1998	Galipeau, K. J., et al.	
		5802106	Sep 1, 1998	Packer, R. L.	
		5802291	Sep 1, 1998	Balick, M. et al.	
		5812769	Sep 22, 1998	Graber, T. E., et al.	
		5815664	Sep 29, 1998	Asano, K.	
		5819092	Oct 6, 1998	Ferguson, Charles H., et al.	
		5828847	Oct 27, 1998	Gehr, C. R., et al.	
		5832069	~~Nov 3, 2000~~	Waters, David B., et al.	11/1998
		5832506	Nov 3, 1998	Kuzma, A. J.	
		5832514	Nov 3, 1998	Norin, S. et al.	
		5835718	Nov 10, 1998	Blewett, C. D.	
		5845090	Dec 1, 1998	Collins, Theodore J., et al.	
		5845303	Dec 1, 1998	Templeman, M. L.	
		5856974	Jan 5, 1999	Gervais, J. L., et al.	
		5862339	Jan 19, 1999	Bonnaure, P. P., et al.	
		5867706	Feb 2, 1999	Martin, Sean J., et al.	
		5867799	Feb 2, 1999	Lang, A. K., et al.	
		5870546	Feb 9, 1999	Kirsch, S. T.	
		5870559	Feb 9, 1999	Leshem, E. et al.	
		5878212	Mar 2, 1999	Civanlar, S. et al.	
		5884038	Mar 16, 1999	Kapoor, R.	
		5890171	Mar 30, 1999	Blumer, T. P., et al.	
		5893116	Apr 6, 1999	Simmonds, C. D., et al.	
		5894554	Apr 13, 1999	Lowery, K. et al.	
		5896533	Apr 20, 1999	Ramos, D. et al.	
		5898456	Apr 27, 1999	Wahl, S.	
		5903723	May 11, 1999	Beck, R. D., et al.	
		5907704	May 25, 1999	Gudmundson, N. K., et al.	

EXAMINER	/Thomas Dailey/	**DATE CONSIDERED**	10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

PTO/SB/08a (01-08)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number..

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David FULLAGAR
Art Unit	2452
Examiner Name	Neeraj K. Utreja

Sheet	2	of	2	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		7185052B2	Feb 27, 2007	Day, R	
		7254645	Aug 7, 2007	Nishi, Koji	
		7274658	Sep 25, 2007	Bornstein, Claudson F., et al.	
		7359955	Apr 15, 2008	Menon, Satish N., et al.	
		7373644B2	May 13, 2008	Aborn, J.	
		7562153	Jul 14, 2009	Biliris, Alexandros et al.	
		7577754B2	Aug 18, 2009	~~Garcia, L. A., et al.~~ Garcia-Luna-Aceves et al.	

Change(s) applied to document,
/R.K.C./
12/5/2014

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No	Foreign Patent Document	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	T[2]

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No [1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]

EXAMINER	/Neeraj Utreja/	DATE CONSIDERED	11/10/2011

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /N.U

PTO/SB/08a (01-08)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number..

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David FULLAGAR
Art Unit	2452
Examiner Name	Neeraj K. Utreja

Sheet	1	of	2	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		20010027491A1	Oct 4, 2001	Terretta, M	
		20020016831	Feb 7, 2002	Peled, Ariel et al.	
		20020016835A1	Feb 7, 2002	Gamerman, G	
		20020040404A1	Apr 4, 2002	Lahr, Nils B.	
		20020087797A1	Jul 4, 2002	Adrangi, F	
		20020116444	Aug 22, 2002	Chaudhri, Imran et al.	
		20020133537A1	Sep 19, 2002	Lau, F	
		20020198953A1	Dec 26, 2002	O'Rourke, B. P., et al.	
		20030031176A1	Feb 13, 2003	Sim, S. Y.	
		20030112792A1	Jun 19, 2003	Cranor, C. D., et al.	
		20030149581	Aug 7, 2003	Chaudhri, Imran et al.	
		20050021863A1	Jan 27, 2005	Jungck, P. J.	
		20060015574	Jan 19, 2006	Seed, Steven L., et al.	
		20080065724	Mar 13, 2008	Seed, Steven L., et al.	
		20080071859	Mar 20, 2008	Seed, Steven L., et al.	
		20080147866	Jun 19, 2008	Stolorz, Paul E., et al.	
		20110145386	Jun 16, 2011	Stolorz, Paul E., et al.	
		5777988	Jul 7, 1998	Cisneros, Arturo	
		5815662	Sep 29, 1998	Ong, L	
		6085193	Jul 4, 2000	Malkin, P. K., et al.	
		6094680	Jul 25, 2009	Hokanson, T. E.	
		6463454	Oct 8, 2002	Lumelsky, Leon et al.	
		6591288	Jul 8, 2003	Edwards, Keith R., et al.	
		6594260B1	Jul 15, 2003	Aviani Jr, J. A., et al.	
		6598121B2	Jul 22, 2003	Challenger, J. R., et al.	
		6732237B1	May 4, 2004	Jacobs, L. et al.	
		6741990	May 25, 2004	Nair, Sandeep R., et al.	
		6785704	Aug 31, 2004	McCanne, Steve	
		6839758	~~Apr 1, 2005~~	Sorensen, Lauge S.	1/2005
		6859840B2	Feb 22, 2005	Singal, S. S., et al.	
		6976090	Dec 13, 2005	Ben-Shaul, Israel et al.	
		6999988	Feb 14, 2006	Buddhikot, Milind M., et al.	
		7024466B2	Apr 4, 2006	Outten, Todd A., et al.	
		7103645B2	Sep 5, 2006	Leighton, F. T., et al.	

Change(s) applied to document, /R.K.C./ 12/5/2014

EXAMINER	/Neeraj Utreja/	**DATE CONSIDERED**	11/10/2011

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /N.U

PTO/SB/08a (01-08)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number..

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	1	of	1	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS					
Examiner Initial *	**Cite No**	**Document Number**	**Publication Date**	**Name of Patentee or Applicant of Cited Document**	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear

FOREIGN PATENT DOCUMENTS						
Examiner Initials*	**Cite No**	**Foreign Patent Document**	**Publication Date**	**Name of Patentee or Applicant of cited Document**	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	**T²**

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No ¹**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T²**
		Canadian Examination Report, dated September 8, 2014, Application No. 2,720,353, filed March 21, 2009; Dkt. No. 0137-CA-01 , 4 pgs.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 11/25/2014

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. ¹ Applicant's unique citation designation number (optional) ² Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	~~Neeraj K. Ureja~~ Thomas Dailey

Sheet	1	of	1	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No	Foreign Patent Document	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	T[2]
		WO-2003088065	Oct 23, 2003	Dilley, John A., et al.		

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No [1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		Supplemental European Search Report, dated November 5, 2012, Application No. 09727342.9, Dkt. No. 0137-EP-01, 7 pgs.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 11/25/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

PTO/SB/08a (01-08)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number..

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	~~Neeraj K. Ureja~~ Thomas Dailey

Sheet	1	of	1	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6157648	Dec 5, 2000	Volt, Eric A., et al.	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No	Foreign Patent Document	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	T[2]
		EP-0817444A2	Jan 7, 1998	Ebrahim, Z.		
		WO-2000/052594A2	Sep 8, 2000	LACHMAN, R. et al.		

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No [1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		, "Extended European Search Report dated December 6, 2011", App. No. 10178695.2, Dkt. No. 0093-EP-C1, , 7 pgs.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 11/25/2014

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

PTO/SB/08a (01-08)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number..

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	~~Neeraj K. Uneja~~ Thomas Dailey

Sheet	1	of	1	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		20040010588A1	Jan 15, 2004	Slater, A et al.	
		20040064832A1	Apr 1, 2004	Tsukidate, R et al.	
		20050060493A1	Mar 17, 2005	Krissell, D et al.	
		20050240966A1	Oct 27, 2005	Hindle, J et al.	
		20100169772A1	Jul 1, 2010	Stallings, H et al.	
		7219153B1	May 15, 2007	Day, M	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No	Foreign Patent Document	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	T[2]

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No [1]	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T[2]
		, "International Search Report, Intl Appl. No. PCT/US11/51252", Intl Filing Date 9/12/2011, Dkt No. 0137-WO-I1 , 3 pgs.	
		, "Written Opinion, Intl Appl No. PCT/US11/51252", Intl Filing Date 9/12/2011, Dkt No. 0137-WO-I1 , 5 pgs.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 11/25/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/408,681	03/21/2009	David Fullagar	0137-US-U1	9965

83579 7590 12/01/2014

LEVEL 3 COMMUNICATIONS, LLC
Attn: Patent Docketing
1025 Eldorado Blvd.
Broomfield, CO 80021

EXAMINER
DAILEY, THOMAS J

ART UNIT	PAPER NUMBER
2452	

NOTIFICATION DATE	DELIVERY MODE
12/01/2014	ELECTRONIC

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Notice of the Office communication was sent electronically on above-indicated "Notification Date" to the following e-mail address(es):

patent.docketing@level3.com

UNITED STATES DEPARTMENT OF COMMERCE
U.S. Patent and Trademark Office
Address : COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450

APPLICATION NO./ CONTROL NO.	FILING DATE	FIRST NAMED INVENTOR / PATENT IN REEXAMINATION	ATTORNEY DOCKET NO.
12/408,681	21 March, 2009	FULLAGAR ET AL.	0137-US-U1

LEVEL 3 COMMUNICATIONS, LLC
Attn: Patent Docketing
1025 Eldorado Blvd.
Broomfield, CO 80021

EXAMINER	
THOMAS DAILEY	
ART UNIT	PAPER
2452	20141125

DATE MAILED:

Please find below and/or attached an Office communication concerning this application or proceeding.

Commissioner for Patents

Attached are the annotated copies of the IDSs submitted 12/20/2011, 2/7/2012, 7/3/2012, 11/14/2012, and 11/6/2014.

Thomas Dailey
AU 2452
571-270-1246

/THOMAS DAILEY/
Primary Examiner, Art Unit 2452

PTO-90C (Rev.04-03)

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	~~Neeraj K. Uteja~~ Thomas Dailey

Sheet	1	of	1	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS					
Examiner Initial *	**Cite No**	**Document Number**	**Publication Date**	**Name of Patentee or Applicant of Cited Document**	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear

FOREIGN PATENT DOCUMENTS						
Examiner Initials*	**Cite No**	**Foreign Patent Document**	**Publication Date**	**Name of Patentee or Applicant of cited Document**	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	T[2]

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	T[2]
		Chinese Examination Report, dated June 13, 2012, Application No. 200980120043.0, Dkt No. 0137-CN-01, 6 pgs.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 11/25/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

83579 7590 11/07/2014
LEVEL 3 COMMUNICATIONS, LLC
Attn: Patent Docketing
1025 Eldorado Blvd.
Broomfield, CO 80021

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

N/A: via EFS-Web	(Depositor's name)
	(Signature)
	(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/408,681	03/21/2009	David Fullagar	0137-US-U1	9965

TITLE OF INVENTION: HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	UNDISCOUNTED	$960	$0	$0	$960	02/09/2015

EXAMINER	ART UNIT	CLASS-SUBCLASS
DAILEY, THOMAS J	2452	709-226000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
- ☐ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
- ☐ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
(1) The names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) The name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 _____
2 _____
3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE: LEVEL 3 COMMUNICATIONS, LLC

(B) RESIDENCE: (CITY and STATE OR COUNTRY): 1025 Eldorado Blvd. Broomfield, CO 80021

Please check the appropriate assignee category or categories (will not be printed on the patent) : ☐ Individual ☒ Corporation or other private group entity ☐ Government

4a. The following fee(s) are submitted:
- ☒ Issue Fee
- ☐ Publication Fee (No small entity discount permitted)
- ☐ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**
- ☐ A check is enclosed.
- ☒ Payment by credit card. XXXXXXXXXXXXXXXX via EFS-Web
- ☐ The Director is hereby authorized to charge the required fee(s), any deficiency, or credits any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)
- ☐ Applicant certifying micro entity status. See 37 CFR 1.29 — NOTE: Absent a valid certification of Micro Entity Status (see forms PTO/SB/15A and 15B), issue fee payment in the micro entity amount will not be accepted at the risk of application abandonment.
- ☐ Applicant asserting small entity status. See 37 CFR 1.27 — NOTE: If the application was previously under micro entity status, checking this box will be taken to be a notification of loss of entitlement to micro entity status.
- ☐ Applicant changing to regular undiscounted fee status. — NOTE: Checking this box will be taken to be a notification of loss of entitlement to small or micro entity status, as applicable.

NOTE: This form must be signed in accordance with 37 CFR 1.31 and 1.33. See 37 CFR 1.4 for signature requirements and certifications.

Authorized Signature /JCS/ Date November 11, 2014

Typed or printed name Jonathan C. Siekmann Registration No. 58259

PTOL-85 Part B (10-13) Approved for use through 10/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re the Application of:

Applicant	: David Fullagar et al.	
Appln. No.	: 12/408,681	Confirmation No: 9965
Filed	: March 21, 2009	Group Art Unit: 2452
Title	: HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)	Examiner: Thomas J. DAILEY Docket No. 0137-US-U1

COMMENTS ON STATEMENT OF REASONS FOR ALLOWANCE

Via EFS-Web

Dear Sir:

Applicant acknowledges the October 7, 2014, and November 7, 2014, Notices of Allowability in the present application by the Examiner with appreciation. Applicant notes that the reasons for allowance set forth in the Notices of Allowability are not the only reasons that the claims are allowable. As the Examiner's reasons for allowance are not exhaustive, such reasons for allowance do not establish an estoppel against Applicant with regard to the scope of any claim in the present application or any related applications, including without limitation any continuation applications claiming priority to the present application.

If there are any questions, the Examiner is invited to contact the undersigned at the telephone number listed below.

Respectfully submitted,

Date: November 11, 2014

/JCS/

Jonathan C. Siekmann, Reg. No. 58,259
Level 3 Communications, LLC
1025 Eldorado Blvd.
Broomfield, Colorado 80021
Tel: 720-888-2140
Fax: 720-888-5619

Electronic Patent Application Fee Transmittal

Application Number:	12408681
Filing Date:	21-Mar-2009
Title of Invention:	HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)
First Named Inventor/Applicant Name:	David Fullagar
Filer:	David Dewey Wier/Jonathan Siekmann
Attorney Docket Number:	0137-US-U1

Filed as Large Entity

Utility under 35 USC 111(a) Filing Fees

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Utility Appl Issue Fee	1501	1	960	960
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous:				
			Total in USD ($)	**960**

Electronic Acknowledgement Receipt

EFS ID:	20668429
Application Number:	12408681
International Application Number:	
Confirmation Number:	9965
Title of Invention:	HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)
First Named Inventor/Applicant Name:	David Fullagar
Customer Number:	83579
Filer:	David Dewey Wier/Jonathan Siekmann
Filer Authorized By:	David Dewey Wier
Attorney Docket Number:	0137-US-U1
Receipt Date:	11-NOV-2014
Filing Date:	21-MAR-2009
Time Stamp:	19:45:33
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	Credit Card
Payment was successfully received in RAM	$960
RAM confirmation Number	14417
Deposit Account	505063
Authorized User	SIEKMANN, JONATHAN

The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:

Charge any Additional Fees required under 37 C.F.R. Section 1.16 (National application filing, search, and examination fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.17 (Patent application and reexamination processing fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.19 (Document supply fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.20 (Post Issuance fees)

Charge any Additional Fees required under 37 C.F.R. Section 1.21 (Miscellaneous fees and charges)

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Issue Fee Payment (PTO-85B)	0137_US_U1_IssueFeeTransmittal.pdf	103081 5f08060b9d6499fcc3c5b8d57dd297b769e2b11f	no	1
Warnings:					
Information:					
2	Post Allowance Communication - Incoming	0137_US_U1_CSRA.pdf	19559 72f67c599d6399acd705baa2d8a52820a7ea9cfe	no	1
Warnings:					
Information:					
3	Fee Worksheet (SB06)	fee-info.pdf	30740 ade2df26a2ee0fe69ad451bb386d6f28da659f79	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	153380		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

<u>New Applications Under 35 U.S.C. 111</u>
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

<u>National Stage of an International Application under 35 U.S.C. 371</u>
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

<u>New International Application Filed with the USPTO as a Receiving Office</u>
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

NOTICE OF ALLOWANCE AND FEE(S) DUE

83579 7590 11/07/2014

LEVEL 3 COMMUNICATIONS, LLC
Attn: Patent Docketing
1025 Eldorado Blvd.
Broomfield, CO 80021

EXAMINER
DAILEY, THOMAS J

ART UNIT	PAPER NUMBER
2452	

DATE MAILED: 11/07/2014

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/408,681	03/21/2009	David Fullagar	0137-US-U1	9965

TITLE OF INVENTION: HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	UNDISCOUNTED	$960	$0	$0	$960	02/09/2015

THE APPLICATION IDENTIFIED ABOVE HAS BEEN EXAMINED AND IS ALLOWED FOR ISSUANCE AS A PATENT. PROSECUTION ON THE MERITS IS CLOSED. THIS NOTICE OF ALLOWANCE IS NOT A GRANT OF PATENT RIGHTS. THIS APPLICATION IS SUBJECT TO WITHDRAWAL FROM ISSUE AT THE INITIATIVE OF THE OFFICE OR UPON PETITION BY THE APPLICANT. SEE 37 CFR 1.313 AND MPEP 1308.

THE ISSUE FEE AND PUBLICATION FEE (IF REQUIRED) MUST BE PAID WITHIN THREE MONTHS FROM THE MAILING DATE OF THIS NOTICE OR THIS APPLICATION SHALL BE REGARDED AS ABANDONED. THIS STATUTORY PERIOD CANNOT BE EXTENDED. SEE 35 U.S.C. 151. THE ISSUE FEE DUE INDICATED ABOVE DOES NOT REFLECT A CREDIT FOR ANY PREVIOUSLY PAID ISSUE FEE IN THIS APPLICATION. IF AN ISSUE FEE HAS PREVIOUSLY BEEN PAID IN THIS APPLICATION (AS SHOWN ABOVE), THE RETURN OF PART B OF THIS FORM WILL BE CONSIDERED A REQUEST TO REAPPLY THE PREVIOUSLY PAID ISSUE FEE TOWARD THE ISSUE FEE NOW DUE.

HOW TO REPLY TO THIS NOTICE:

I. Review the ENTITY STATUS shown above. If the ENTITY STATUS is shown as SMALL or MICRO, verify whether entitlement to that entity status still applies.

If the ENTITY STATUS is the same as shown above, pay the TOTAL FEE(S) DUE shown above.

If the ENTITY STATUS is changed from that shown above, on PART B - FEE(S) TRANSMITTAL, complete section number 5 titled "Change in Entity Status (from status indicated above)".

For purposes of this notice, small entity fees are 1/2 the amount of undiscounted fees, and micro entity fees are 1/2 the amount of small entity fees.

II. PART B - FEE(S) TRANSMITTAL, or its equivalent, must be completed and returned to the United States Patent and Trademark Office (USPTO) with your ISSUE FEE and PUBLICATION FEE (if required). If you are charging the fee(s) to your deposit account, section "4b" of Part B - Fee(s) Transmittal should be completed and an extra copy of the form should be submitted. If an equivalent of Part B is filed, a request to reapply a previously paid issue fee must be clearly made, and delays in processing may occur due to the difficulty in recognizing the paper as an equivalent of Part B.

III. All communications regarding this application must give the application number. Please direct all communications prior to issuance to Mail Stop ISSUE FEE unless advised to the contrary.

IMPORTANT REMINDER: Utility patents issuing on applications filed on or after Dec. 12, 1980 may require payment of maintenance fees. It is patentee's responsibility to ensure timely payment of maintenance fees when due.

PART B - FEE(S) TRANSMITTAL

Complete and send this form, together with applicable fee(s), to: Mail **Mail Stop ISSUE FEE**
Commissioner for Patents
P.O. Box 1450
Alexandria, Virginia 22313-1450
or Fax (571)-273-2885

INSTRUCTIONS: This form should be used for transmitting the ISSUE FEE and PUBLICATION FEE (if required). Blocks 1 through 5 should be completed where appropriate. All further correspondence including the Patent, advance orders and notification of maintenance fees will be mailed to the current correspondence address as indicated unless corrected below or directed otherwise in Block 1, by (a) specifying a new correspondence address; and/or (b) indicating a separate "FEE ADDRESS" for maintenance fee notifications.

CURRENT CORRESPONDENCE ADDRESS (Note: Use Block 1 for any change of address)

83579 7590 11/07/2014
LEVEL 3 COMMUNICATIONS, LLC
Attn: Patent Docketing
1025 Eldorado Blvd.
Broomfield, CO 80021

Note: A certificate of mailing can only be used for domestic mailings of the Fee(s) Transmittal. This certificate cannot be used for any other accompanying papers. Each additional paper, such as an assignment or formal drawing, must have its own certificate of mailing or transmission.

Certificate of Mailing or Transmission
I hereby certify that this Fee(s) Transmittal is being deposited with the United States Postal Service with sufficient postage for first class mail in an envelope addressed to the Mail Stop ISSUE FEE address above, or being facsimile transmitted to the USPTO (571) 273-2885, on the date indicated below.

(Depositor's name)
(Signature)
(Date)

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/408,681	03/21/2009	David Fullagar	0137-US-U1	9965

TITLE OF INVENTION: HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)

APPLN. TYPE	ENTITY STATUS	ISSUE FEE DUE	PUBLICATION FEE DUE	PREV. PAID ISSUE FEE	TOTAL FEE(S) DUE	DATE DUE
nonprovisional	UNDISCOUNTED	$960	$0	$0	$960	02/09/2015

EXAMINER	ART UNIT	CLASS-SUBCLASS
DAILEY, THOMAS J	2452	709-226000

1. Change of correspondence address or indication of "Fee Address" (37 CFR 1.363).
❑ Change of correspondence address (or Change of Correspondence Address form PTO/SB/122) attached.
❑ "Fee Address" indication (or "Fee Address" Indication form PTO/SB/47; Rev 03-02 or more recent) attached. **Use of a Customer Number is required.**

2. For printing on the patent front page, list
(1) The names of up to 3 registered patent attorneys or agents OR, alternatively,
(2) The name of a single firm (having as a member a registered attorney or agent) and the names of up to 2 registered patent attorneys or agents. If no name is listed, no name will be printed.
1 _____
2 _____
3 _____

3. ASSIGNEE NAME AND RESIDENCE DATA TO BE PRINTED ON THE PATENT (print or type)

PLEASE NOTE: Unless an assignee is identified below, no assignee data will appear on the patent. If an assignee is identified below, the document has been filed for recordation as set forth in 37 CFR 3.11. Completion of this form is NOT a substitute for filing an assignment.

(A) NAME OF ASSIGNEE (B) RESIDENCE: (CITY and STATE OR COUNTRY)

Please check the appropriate assignee category or categories (will not be printed on the patent) : ❑ Individual ❑ Corporation or other private group entity ❑ Government

4a. The following fee(s) are submitted:
❑ Issue Fee
❑ Publication Fee (No small entity discount permitted)
❑ Advance Order - # of Copies _____

4b. Payment of Fee(s): **(Please first reapply any previously paid issue fee shown above)**
❑ A check is enclosed.
❑ Payment by credit card. Form PTO-2038 is attached.
❑ The Director is hereby authorized to charge the required fee(s), any deficiency, or credits any overpayment, to Deposit Account Number _____ (enclose an extra copy of this form).

5. **Change in Entity Status** (from status indicated above)
❑ Applicant certifying micro entity status. See 37 CFR 1.29
NOTE: Absent a valid certification of Micro Entity Status (see forms PTO/SB/15A and 15B), issue fee payment in the micro entity amount will not be accepted at the risk of application abandonment.
❑ Applicant asserting small entity status. See 37 CFR 1.27
NOTE: If the application was previously under micro entity status, checking this box will be taken to be a notification of loss of entitlement to micro entity status.
❑ Applicant changing to regular undiscounted fee status.
NOTE: Checking this box will be taken to be a notification of loss of entitlement to small or micro entity status, as applicable.

NOTE: This form must be signed in accordance with 37 CFR 1.31 and 1.33. See 37 CFR 1.4 for signature requirements and certifications.

Authorized Signature _____ Date _____

Typed or printed name _____ Registration No. _____

PTOL-85 Part B (10-13) Approved for use through 10/31/2013. OMB 0651-0033 U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
12/408,681	03/21/2009	David Fullagar	0137-US-U1	9965

83579 7590 11/07/2014

LEVEL 3 COMMUNICATIONS, LLC
Attn: Patent Docketing
1025 Eldorado Blvd.
Broomfield, CO 80021

EXAMINER	
DAILEY, THOMAS J	
ART UNIT	PAPER NUMBER
2452	

DATE MAILED: 11/07/2014

Determination of Patent Term Adjustment under 35 U.S.C. 154 (b)
(Applications filed on or after May 29, 2000)

The Office has discontinued providing a Patent Term Adjustment (PTA) calculation with the Notice of Allowance.

Section 1(h)(2) of the AIA Technical Corrections Act amended 35 U.S.C. 154(b)(3)(B)(i) to eliminate the requirement that the Office provide a patent term adjustment determination with the notice of allowance. See Revisions to Patent Term Adjustment, 78 Fed. Reg. 19416, 19417 (Apr. 1, 2013). Therefore, the Office is no longer providing an initial patent term adjustment determination with the notice of allowance. The Office will continue to provide a patent term adjustment determination with the Issue Notification Letter that is mailed to applicant approximately three weeks prior to the issue date of the patent, and will include the patent term adjustment on the patent. Any request for reconsideration of the patent term adjustment determination (or reinstatement of patent term adjustment) should follow the process outlined in 37 CFR 1.705.

Any questions regarding the Patent Term Extension or Adjustment determination should be directed to the Office of Patent Legal Administration at (571)-272-7702. Questions relating to issue and publication fee payments should be directed to the Customer Service Center of the Office of Patent Publication at 1-(888)-786-0101 or (571)-272-4200.

OMB Clearance and PRA Burden Statement for PTOL-85 Part B

The Paperwork Reduction Act (PRA) of 1995 requires Federal agencies to obtain Office of Management and Budget approval before requesting most types of information from the public. When OMB approves an agency request to collect information from the public, OMB (i) provides a valid OMB Control Number and expiration date for the agency to display on the instrument that will be used to collect the information and (ii) requires the agency to inform the public about the OMB Control Number's legal significance in accordance with 5 CFR 1320.5(b).

The information collected by PTOL-85 Part B is required by 37 CFR 1.311. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, Virginia 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, Virginia 22313-1450. Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the information solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether disclosure of these records is required by the Freedom of Information Act.
2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.
3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.
4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).
5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.
6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).
7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.
8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record may be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspection or an issued patent.
9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

Notice of Allowability	Application No. 12/408,681	Applicant(s) FULLAGAR ET AL.	
	Examiner THOMAS DAILEY	Art Unit 2452	AIA (First Inventor to File) Status No

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address--

All claims being allowable, PROSECUTION ON THE MERITS IS (OR REMAINS) CLOSED in this application. If not included herewith (or previously mailed), a Notice of Allowance (PTOL-85) or other appropriate communication will be mailed in due course. **THIS NOTICE OF ALLOWABILITY IS NOT A GRANT OF PATENT RIGHTS.** This application is subject to withdrawal from issue at the initiative of the Office or upon petition by the applicant. See 37 CFR 1.313 and MPEP 1308.

1. ☒ This communication is responsive to *RCE entered 10/21/2014*.
   ☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on_____.
2. ☐ An election was made by the applicant in response to a restriction requirement set forth during the interview on _____; the restriction requirement and election have been incorporated into this action.
3. ☒ The allowed claim(s) is/are *1-29*. As a result of the allowed claim(s), you may be eligible to benefit from the **Patent Prosecution Highway** program at a participating intellectual property office for the corresponding application. For more information, please see http://www.uspto.gov/patents/init_events/pph/index.jsp or send an inquiry to PPHfeedback@uspto.gov .
4. ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

   Certified copies:

   a) ☐ All b) ☐ Some *c) ☐ None of the:
   1. ☐ Certified copies of the priority documents have been received.
   2. ☐ Certified copies of the priority documents have been received in Application No. _____ .
   3. ☐ Copies of the certified copies of the priority documents have been received in this national stage application from the International Bureau (PCT Rule 17.2(a)).

   * Certified copies not received: _____.

Applicant has THREE MONTHS FROM THE "MAILING DATE" of this communication to file a reply complying with the requirements noted below. Failure to timely comply will result in ABANDONMENT of this application.
THIS THREE-MONTH PERIOD IS NOT EXTENDABLE.

5. ☐ CORRECTED DRAWINGS (as "replacement sheets") must be submitted.
   ☐ including changes required by the attached Examiner's Amendment / Comment or in the Office action of Paper No./Mail Date _____.

   Identifying indicia such as the application number (see 37 CFR 1.84(c)) should be written on the drawings in the front (not the back) of each sheet. Replacement sheet(s) should be labeled as such in the header according to 37 CFR 1.121(d).
6. ☐ DEPOSIT OF and/or INFORMATION about the deposit of BIOLOGICAL MATERIAL must be submitted. Note the attached Examiner's comment regarding REQUIREMENT FOR THE DEPOSIT OF BIOLOGICAL MATERIAL.

Attachment(s)

1. ☐ Notice of References Cited (PTO-892)
2. ☒ Information Disclosure Statements (PTO/SB/08), Paper No./Mail Date _____
3. ☐ Examiner's Comment Regarding Requirement for Deposit of Biological Material
4. ☐ Interview Summary (PTO-413), Paper No./Mail Date _____ .
5. ☐ Examiner's Amendment/Comment
6. ☐ Examiner's Statement of Reasons for Allowance
7. ☐ Other _____.

/THOMAS DAILEY/
Primary Examiner, Art Unit 2452

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	1	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		20010029525	Oct 11, 2001	Lahr, Nils B.	
		20010049732A1	Dec 6, 2001	Raciborski, N. F., et al.	
		20010051980	Dec 13, 2001	Raciborski, Nathan F., et al.	
		20010056500	Dec 27, 2001	Farber, David A., et al.	
		20020018449	Feb 14, 2002	Ricciulli, Livio	
		20020023164	Feb 14, 2002	Lahr, Nils B.	
		20020023165	Feb 21, 2002	Lahr, Nils B.	
		20020042817	Apr 11, 2002	Lahr, Nils B.	
		20020046273	Apr 18, 2002	Lahr, Nils B., et al.	
		20020046405	Apr 18, 2002	Lahr, Nils B.	
		20020049857	Apr 25, 2002	Farber, David A., et al.	
		20020059592	May 16, 2002	Kiraly, Jozsef	
		20020066038	May 30, 2002	Mattsson, Ulf	
		20020073199	Jun 13, 2002	Levine, Matthew et al.	
		20020078233	Jun 20, 2002	Biliris, Alexandros et al.	
		20020082999	Jun 27, 2002	Lee, Cheol-Woong et al.	
		20020083124	Jun 27, 2002	Knox, Christopher R., et al.	
		20020089470	Jul 11, 2002	Raman, Mithila et al.	
		20020092026	Jul 11, 2002	Janniello, James P., et al.	
		20020099850	Jul 25, 2002	Farber, David A., et al.	
		20020124080	Sep 5, 2002	Leighton, F. T.	
		20020129134	Sep 12, 2002	Leighton, F. T., et al.	
		20020131645	Sep 19, 2002	Hamilton, Chris H.	
		20020143798	Oct 3, 2002	Lisiecki, Philip A., et al.	
		20020143888	Oct 3, 2002	Lisiecki, Philip A., et al.	
		20020147774	Oct 10, 2002	Lisiecki, Philip A., et al.	
		20020152318	Oct 17, 2002	Menon, Satish N., et al.	
		20020163882	Nov 7, 2002	Bornstein, Claudson F., et al.	
		20020199016	Dec 26, 2002	Freedman, Avraham T.	
		20030009444	Jan 9, 2003	Eidler, Christopher W., et al.	
		20030018966	Jan 23, 2003	Cook, David H., et al.	
		20030028623	Feb 6, 2003	Hennessey, Wade L., et al.	
		20030028626	Feb 6, 2003	Hennessey, Wade L., et al.	
		20030028777	Feb 6, 2003	Hennessey, Wade L., et al.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	2	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		20030055972	Mar 20, 2003	Fuller, William T., et al.	
		20030061263	Mar 27, 2003	Riddle, Guy	
		20030061280	Mar 27, 2003	Bulson, Jason A., et al.	
		20030065703	Apr 3, 2003	Aborn, Justin	
		20030065762	Apr 3, 2003	Stolorz, Paul E., et al.	
		20030078888	Apr 24, 2003	Lee, Cheol-Woong et al.	
		20030078889	Apr 24, 2003	Lee, Cheol-Woong et al.	
		20030079027	Apr 24, 2003	Slocombe, Michael et al.	
		20030095660	May 22, 2003	Lee, Chang Y., et al.	
		20030105604	Jun 5, 2003	Ash, Leslie E., et al.	
		20030154239	Aug 14, 2003	Davis, Andrew T., et al.	
		20030158923	Aug 21, 2003	Burkhart, Reed	
		20030158928	Aug 21, 2003	Knox, Christopher R., et al.	
		20040022194	Feb 5, 2004	Ricciulli, Livio	
		20040139097	Jul 15, 2004	Farber, David A., et al.	
		20040177148	Sep 9, 2004	Tsimelzon Jr., Mark	
		20050033858	Feb 10, 2005	Swildens, Eric S., et al.	
		20050038851	Feb 17, 2005	Kriegsman, Mark E.	
		20050100027	May 12, 2005	Leinwand, Allan S., et al.	
		20050114296	May 26, 2005	Farber, David A., et al.	
		20050262104	Nov 24, 2005	Robertson, Niel et al.	
		20060143293	Jun 29, 2006	Freedman, Avraham T.	
		4495570	Jan 22, 1985	Kitajima, Hiroyuki et al.	
		4591983	May 27, 1986	Bennett, James S., et al.	
		4594704	Jun 10, 1986	Ollivier, Y.	
		4726017	Feb 16, 1988	Krum, H. et al.	
		4803641	Feb 7, 1989	Hardy, S. et al.	
		4839798	Jun 13, 1989	Eguchi, H. et al.	
		4847784	Jul 11, 1989	Clancey, W. J.	
		4920432	Apr 24, 1990	Eggers, D. et al.	
		4922417	May 1, 1990	Churm, B. et al.	
		4943932	Jul 24, 1990	Lark, J. S., et al.	
		4949187	Aug 14, 1990	Cohen, J. M.	
		4949248	Aug 14, 1990	Caro, M. A.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	3	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5029232	Jul 2, 1991	Nall, D. C.	
		5130792	Jul 14, 1992	Tindell, E. G., et al.	
		5132992	Jul 21, 1992	Yurt, P. et al.	
		5136716	Aug 4, 1992	Harvey, G. et al.	
		5172413	Dec 15, 1992	Bradley, G. C., et al.	
		5191573	Mar 2, 1993	Hair, A. R.	
		5253275	Oct 12, 1993	Yurt, P. et al.	
		5253341	Oct 12, 1993	Rozmanith, A. et al.	
		5287499	Feb 15, 1994	Nemes, R. M.	
		5287537	Feb 15, 1994	Newmark, R. J., et al.	
		5291554	Mar 1, 1994	Morales, F.	
		5341477	Aug 23, 1994	Pitkin, R. P., et al.	
		5371532	Dec 6, 1994	Gelman, A. et al.	
		5410343	Apr 25, 1995	Coddington, C. D., et al.	
		5414455	May 9, 1995	Hooper, D. F., et al.	
		5442389	Aug 15, 1995	Blahut, D. E., et al.	
		5442390	Aug 15, 1995	Hooper, D. F., et al.	
		5442749	Aug 15, 1995	Northcutt, J. D., et al.	
		5471622	Nov 28, 1995	Eadline, D. J.	
		5475615	Dec 12, 1995	Lin, K. D.	
		5508732	Apr 16, 1996	Bottomley, J. F., et al.	
		5515511	May 7, 1996	Nguyen, T. et al.	
		5519435	May 21, 1996	Anderson, M. H.	
		5528281	Jun 18, 1996	Grady, J. et al.	
		5539621	Jul 23, 1996	Kikinis, D.	
		5542087	Jul 30, 1996	Neimat, M. et al.	
		5544313	Aug 6, 1996	Shachnai, H. et al.	
		5544327	Aug 6, 1996	Dan, A. et al.	
		5550577	Aug 27, 1996	Verbiest, W. J., et al.	
		5550863	Aug 27, 1996	Yurt, P. et al.	
		5550982	Aug 27, 1996	Long, J. E., et al.	
		5557317	Sep 17, 1996	Nishio, M. et al.	
		5568181	Oct 22, 1998	Greenwood, David G., et al.	
		5572643	Nov 5, 1996	Judson, D. H.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	4	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5590288	Dec 31, 1996	Castor, P. F., et al.	
		5592611	Jan 7, 1997	Midgely, C. W., et al.	
		5594910	Jan 14, 1997	Filepp, R. et al.	
		5603026	Feb 11, 1997	Demers, A. J., et al.	
		5614940	Mar 25, 1997	Cobbley, David A., et al.	
		5619648	Apr 8, 1997	Canale, L. M., et al.	
		5623656	Apr 22, 1997	Lyons, K. B.	
		5625781	Apr 29, 1997	Cline, T. L., et al.	
		5627829	May 6, 1997	Gleeson, B. J., et al.	
		5630067	May 13, 1997	Kindell, C. N., et al.	
		5633999	May 27, 1997	Clowes, R. F., et al.	
		5634006	May 27, 1997	Baugher, M J., et al.	
		5638443	Jun 10, 1997	Stefik, M. J., et al.	
		5644714	Jul 1, 1997	Kikinis, D.	
		5646676	Jul 8, 1997	Dewkett, T. J., et al.	
		5649186	Jul 15, 1997	Ferguson, G. J.	
		5659729	Aug 19, 1997	Nielsen, J.	
		5666362	Sep 9, 1997	Chen, C. T., et al.	
		5671279	Sep 23, 1997	Elgamal, T.	
		5675734	Oct 7, 1997	Hair, A. R.	
		5682512	Oct 28, 1997	Tetrick, R. S.	
		5699513	Dec 16, 1997	Feigen, R. et al.	
		5712979	Jan 27, 1998	Graber, T. E., et al.	
		5715453	Feb 3, 1998	Stewart, G.	
		5721914	Feb 24, 1998	DeVries, L. R.	
		5734719	Mar 31, 1998	Tsevdos, James T., et al.	
		5734831	Mar 31, 1998	Sanders, J. B.	
		5740423	Apr 14, 1998	Logan, J. R., et al.	
		5742762	Apr 21, 1998	Scholl, Thomas H., et al.	
		5751961	May 12, 1998	Smyk, D.	
		5761507	Jun 2, 1998	Govett, I. R.	
		5761663	Jun 2, 1998	Lagarde, K. C., et al.	
		5764906	Jun 9, 1998	Edelstein, M. et al.	
		5774660	Jun 30, 1998	Brendel, J. et al.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	5	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5774668	Jun 30, 1998	Choquier, P. et al.	
		5777989	Jul 7, 1998	McGarvey, J.	
		5778187	Jul 7, 1998	Monteiro, Antonio M., et al.	
		5781909	Jul 14, 2009	Logan, James et al.	
		5784058	Jul 21, 1998	LaStrange, T. E., et al.	
		5796952	Aug 18, 1998	Davis, O. et al.	
		5799141	Aug 25, 1998	Galipeau, K. J., et al.	
		5802106	Sep 1, 1998	Packer, R. L.	
		5802291	Sep 1, 1998	Balick, M. et al.	
		5812769	Sep 22, 1998	Graber, T. E., et al.	
		5815664	Sep 29, 1998	Asano, K.	
		5819092	Oct 6, 1998	Ferguson, Charles H., et al.	
		5828847	Oct 27, 1998	Gehr, C. R., et al.	
		5832069	Nov 3, 2009	Waters, David B., et al.	
		5832506	Nov 3, 1998	Kuzma, A. J.	
		5832514	Nov 3, 1998	Norin, S. et al.	
		5835718	Nov 10, 1998	Blewett, C. D.	
		5845090	Dec 1, 1998	Collins, Theodore J., et al.	
		5845303	Dec 1, 1998	Templeman, M. L.	
		5856974	Jan 5, 1999	Gervais, J. L., et al.	
		5862339	Jan 19, 1999	Bonnaure, P. P., et al.	
		5867706	Feb 2, 1999	Martin, Sean J., et al.	
		5867799	Feb 2, 1999	Lang, A. K., et al.	
		5870546	Feb 9, 1999	Kirsch, S. T.	
		5870559	Feb 9, 1999	Leshem, E. et al.	
		5878212	Mar 2, 1999	Civanlar, S. et al.	
		5884038	Mar 16, 1999	Kapoor, R.	
		5890171	Mar 30, 1999	Blumer, T. P., et al.	
		5893116	Apr 6, 1999	Simmonds, C. D., et al.	
		5894554	Apr 13, 1999	Lowery, K. et al.	
		5896533	Apr 20, 1999	Ramos, D. et al.	
		5898456	Apr 27, 1999	Wahl, S.	
		5903723	May 11, 1999	Beck, R. D., et al.	
		5907704	May 25, 1999	Gudmundson, N. K., et al.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	6	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5913028	Jun 15, 1999	Wang, F. et al.	
		5913033	Jun 15, 1999	Grout, G. B.	
		5918010	Jun 29, 1999	Appleman, K. H., et al.	
		5919247	Jul 6, 1999	Van, H. A., et al.	
		5920701	Jul 6, 1999	Miller, C. K., et al.	
		5931904	Aug 3, 1999	Banga, Gaurav	
		5933832	Aug 3, 1999	Suzuoka, T. et al.	
		5933835	Aug 3, 1999	Adams, Robert et al.	
		5935207	Aug 10, 1999	Logue, J. D., et al.	
		5940831	Aug 17, 1999	Takano, H.	
		5944780	Aug 31, 1999	Chase, J. S.	
		5951694	Sep 14, 1999	Choquier, P.	
		5956489	Sep 21, 1999	San Andres, R. J., et al.	
		5956716	Sep 21, 1999	Kenner, B. et al.	
		5958008	Sep 28, 1999	Pogrebisky, M. et al.	
		5961596	Oct 5, 1999	Takubo, S. et al.	
		5966440	Oct 12, 1999	Hair, A. R.	
		5968121	Oct 19, 1999	Logan, K. W., et al.	
		5973696	Oct 26, 1999	Agranat, I. D., et al.	
		5978791	Nov 2, 1999	Farber, D. A., et al.	
		5983214	Nov 9, 1999	Lang, A. K., et al.	
		5983227	Nov 9, 1999	Nazem, F. et al.	
		5987606	Nov 16, 1999	Cirasole, P. et al.	
		5991809	Nov 23, 1999	Kriegsman, M. E., et al.	
		5996025	Nov 30, 1999	Day	
		6002720	Dec 14, 1999	Yurt, P. et al.	
		6003030	Dec 14, 1999	Kenner, B. et al.	
		6012090	Jan 4, 2000	Chung, Pi-Yu. et al.	
		6014686	Jan 11, 2000	Elnozahy, E. N., et al.	
		6014698	Jan 11, 2000	Griffiths, M. J.	
		6016509	Jan 18, 2000	Dedrick, Rick	
		6016512	Jan 18, 2000	Huitema, Christian	
		6018516	Jan 25, 2000	Packer, R. L.	
		6021426	Feb 1, 2000	Douglis, F.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	7	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6023470	Feb 8, 2000	Lee, Warren S., et al.	
		6029175	Feb 22, 2000	Chow, Yen-whei et al.	
		6029176	Feb 22, 2000	Cannon, M. E.	
		6035332	Mar 7, 2000	Ingrassia, M. I., et al.	
		6038216	Mar 14, 2000	Packer, R. L.	
		6038310	Mar 14, 2000	Hollywood, S. et al.	
		6041324	Mar 21, 2000	Earl, J. R., et al.	
		6044405	Mar 28, 2000	Driscoll, E. J., et al.	
		6046980	Apr 4, 2000	Packer, R. L.	
		6049831	Apr 11, 2000	Gardell, S. E., et al.	
		6052718	Apr 18, 2000	Gifford, D. K.	
		6052730	Apr 18, 2000	Felciano, R. M., et al.	
		6065051	May 16, 2000	Steele, D. W., et al.	
		6065062	May 16, 2000	Periasamy, R. et al.	
		6070191	May 30, 2000	Narendran, B. et al.	
		6078943	Jun 20, 2000	Yu, P. S.	
		6081829	Jun 27, 2000	Sidana, A. S.	
		6081835	Jun 27, 2000	Antcliff	
		6081840	Jun 27, 2000	Zhao, Yan	
		6092112	Jul 18, 2000	Fukushige, Y.	
		6092204	Jul 18, 2000	Baker, B. S., et al.	
		6094706	Jul 25, 2000	Factor, Michael E., et al.	
		6098078	Aug 1, 2000	Gehani, N. H.	
		6098096	Aug 1, 2000	Tsirigotis, Panagiotis et al.	
		6105028	Aug 15, 2000	Sullivan, D. J., et al.	
		6108673	Aug 22, 2000	Brandt, M. L., et al.	
		6108703	Aug 22, 2000	Leighton, F. T., et al.	
		6112231	Aug 29, 2000	DeSimone, A. et al.	
		6112239	Aug 29, 2000	Kenner, B. et al.	
		6112240	Aug 29, 2000	Pogue, M. A., et al.	
		6115357	Sep 5, 2000	Packer, R. L., et al.	
		6115752	Sep 5, 2000	Chauhan, A.	
		6119143	Sep 12, 2000	Dias, D. M., et al.	
		6125388	Sep 26, 2000	Reisman, R. R.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	8	of	21

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6125394	Sep 26, 2000	Rabinovich, Michael	
		6128601	Oct 3, 2000	Van Horne, P. et al.	
		6128623	Oct 3, 2000	Mattis, Peter et al.	
		6128660	Oct 3, 2000	Grimm, S. M., et al.	
		6130890	Oct 10, 2000	Leinwand, A. S., et al.	
		6131095	Oct 10, 2000	Low, Colin et al.	
		6134583	Oct 17, 2000	Herriot, R. G.	
		6144375	Nov 7, 2000	Jain, R. C., et al.	
		6144702	Nov 7, 2000	Yurt, P. et al.	
		6151624	Nov 21, 2000	Teare, K. et al.	
		6154744	Nov 28, 2000	Kenner, B. et al.	
		6154753	Nov 28, 2000	McFarland, J. A.	
		6161137	Dec 12, 2000	Johnson, Frank E., et al.	
		6163779	Dec 19, 2000	Mantha, M. L., et al.	
		6173311	Jan 9, 2001	Hassett, G. P., et al.	
		6173322	Jan 9, 2001	Hu, W. M.	
		6178160	Jan 23, 2001	Bolton, D. W., et al.	
		6181867	Jan 30, 2001	Kenner, B. et al.	
		6185598	Feb 6, 2001	Farber, David A., et al.	
		6185619	Feb 6, 2001	Joffe, R. L., et al.	
		6189030	Feb 13, 2001	Kirsch, S. T., et al.	
		6189039	Feb 13, 2001	Harvey	
		6195680	Feb 27, 2001	Goldszmidt, German S., et al.	
		6205120	Mar 20, 2001	Packer, R. L., et al.	
		6226618	May 1, 2001	Downs	
		6226642	May 1, 2001	Beranek, M. J., et al.	
		6230196	May 8, 2001	Guenthner, T. J., et al.	
		6233623	May 15, 2001	Jeffords, Jason et al.	
		6240462	May 29, 2001	Agraharam, Sanjay et al.	
		6249810	Jun 19, 2001	Kiraly, J.	
		6256675	Jul 3, 2001	Rabinovich, M.	
		6263313	Jul 17, 2001	Milsted	
		6266335	Jul 24, 2001	Bhaskaran, Sajit	
		6266699	Jul 24, 2001	Sevcik, M.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	9	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6269394	Jul 31, 2001	Kenner, B. et al.	
		6272566	Aug 7, 2001	Craft	
		6275470	Aug 14, 2001	Ricciulli, L.	
		6282569	Aug 28, 2001	Wallis, G. D., et al.	
		6286045	Sep 4, 2001	Griffith, M. J., et al.	
		6298041	Oct 2, 2001	Packer, R. L.	
		6314565	Nov 6, 2001	Kenner, B. et al.	
		6317787	Nov 13, 2001	Boyd, William G., et al.	
		6324580	Nov 27, 2001	Jindal, Anita et al.	
		6324582	Nov 27, 2001	Sridhar, Manickam R., et al.	
		6330602	Dec 11, 2001	Law, E. K., et al.	
		6332157	Dec 18, 2001	Mighdoll, Lee S., et al.	
		6338044	Jan 8, 2002	Cook, D. et al.	
		6343298	Jan 29, 2002	Savchenko, Alexander V., et al.	
		6345294	Feb 5, 2002	O'Toole, James et al.	
		6347085	Feb 12, 2002	Kelly, K. C., et al.	
		6351775	Feb 26, 2002	Yu, P. S.	
		6351776	Feb 26, 2002	O'Brien, Brett et al.	
		6360256	Mar 19, 2002	Lim, S. B.	
		6370571	Apr 9, 2002	Medin, M. S.	
		6370580	Apr 9, 2002	Kriegsman, M. E.	
		6389462	May 14, 2002	Cohen, Ariel et al.	
		6398245	Jun 4, 2002	Gruse	
		6405252	Jun 11, 2002	Gupta, A. K., et al.	
		6405257	Jun 11, 2002	Gersht, A. et al.	
		6412000	Jun 25, 2002	Riddle, G. et al.	
		6415280	Jul 2, 2002	Farber, D. A., et al.	
		6415368	Jul 2, 2002	Glance, Natalie S., et al.	
		6418421	Jul 9, 2002	Hurtado	
		6421714	Jul 16, 2002	Rai, Girish et al.	
		6421726	Jul 16, 2002	Kenner, B. et al.	
		6427170	Jul 30, 2002	Sitaraman, Aravind et al.	
		6430618	Aug 6, 2002	Karger, D. et al.	
		6442588	Aug 27, 2002	Clark, Neal H., et al.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	10	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6453319	Sep 17, 2002	Mattis, Peter et al.	
		6456630	Sep 24, 2002	Packer, R. L., et al.	
		6460082	Oct 1, 2002	Lumelsky	
		6460085	Oct 1, 2002	Toporek, J. D., et al.	
		6463508	Oct 8, 2002	Wolf	
		6466949	Oct 15, 2002	Yang, Mike et al.	
		6470389	Oct 22, 2002	Chung, Pi-Yu et al.	
		6473405	Oct 29, 2002	Ricciulli, L.	
		6480893	Nov 12, 2002	Kriegsman, M. E.	
		6484143	Nov 19, 2002	Swildens, E. S., et al.	
		6484204	Nov 19, 2002	Rabinovich, M.	
		6487555	Nov 26, 2002	Bharat, Krishna A., et al.	
		6490580	Dec 3, 2002	Dey, J. K., et al.	
		6493707	Dec 10, 2002	Dey, J. K., et al.	
		6496856	Dec 17, 2002	Kenner, B. et al.	
		6502125	Dec 31, 2002	Kenner, B. et al.	
		6502205	Dec 31, 2002	Yanai, Moshe et al.	
		6502215	Dec 31, 2002	Raad, G. B., et al.	
		6505248	Jan 7, 2003	Casper, B. J., et al.	
		6529477	Mar 4, 2003	Toporek, J. D., et al.	
		6553413	Apr 22, 2003	Leighton, F T., et al.	
		6553420	Apr 22, 2003	Karger, D. et al.	
		6557054	Apr 29, 2003	Reisman, R. R.	
		6564251	May 13, 2003	Katariya, S. et al.	
		6577595	Jun 10, 2003	Counterman, R. C.	
		6581090	Jun 17, 2003	Lindbo, S. et al.	
		6584083	Jun 24, 2003	Toporek, J. D., et al.	
		6587837	Jul 1, 2003	Spagna	
		6591299	Jul 8, 2003	Riddle, G. et al.	
		6611862	Aug 26, 2003	Reisman, R. R.	
		6625643	Sep 23, 2003	Colby, K. W., et al.	
		6651141	Nov 18, 2003	Adrangi, Farid	
		6654344	Nov 25, 2003	Toporek, J. D., et al.	
		6654807	Nov 25, 2003	Farber, David A., et al.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	11	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6658464	Dec 2, 2003	Reisman, R. R.	
		6665706	Dec 16, 2003	Kenner, B. et al.	
		6678659	Jan 13, 2004	Van Kommer, Robert	
		6691148	Feb 10, 2004	Zinky, J. A., et al.	
		6694358	Feb 17, 2004	Swildens, E. S., et al.	
		6699418	Mar 2, 2004	Okada, K. et al.	
		6708137	Mar 16, 2004	Carley, S.	
		6718328	Apr 6, 2004	Norris, C. L.	
		6741563	May 25, 2004	Packer, R. L.	
		6751673	Jun 15, 2004	Shaw, D. M.	
		6754699	Jun 22, 2004	Swildens, E. S., et al.	
		6754706	Jun 22, 2004	Swildens, E. S., et al.	
		6763377	Jul 13, 2004	Belknap	
		6763388	Jul 13, 2004	Tsimelzon, M.	
		6778502	Aug 17, 2004	Ricciulli, L.	
		6799221	Sep 28, 2004	Kenner, B. et al.	
		6801576	Oct 5, 2004	Haldeman, R. M., et al.	
		6834306	Dec 21, 2004	Tsimelzon, M.	
		6842604	Jan 11, 2005	Cook, D. et al.	
		6859791	Feb 22, 2005	Spagna	
		6870851	Mar 22, 2005	Leinwand, A. S., et al.	
		6874032	Mar 29, 2005	Gersht, A. et al.	
		6901604	May 31, 2005	Kiraly, J.	
		6915307	Jul 5, 2005	Mattis, Peter et al.	
		6915329	Jul 5, 2005	Kriegsman, M. E.	
		6928442	Aug 9, 2005	Farber, D. A., et al.	
		6934255	Aug 23, 2005	Toporek, J. D., et al.	
		6950623	Sep 27, 2005	Brown, J. W., et al.	
		6963910	Nov 8, 2005	Belknap, William R., et al.	
		6963980	Nov 8, 2005	Mattsson, U.	
		6963981	Nov 8, 2005	Bailey, R A., et al.	
		6970432	Nov 29, 2005	Hankins, S. A., et al.	
		6973490	Dec 6, 2005	Robertson, N. et al.	
		6981050	Dec 27, 2005	Tobias, M. et al.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	12	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6981180	Dec 27, 2005	Bailey, S. T., et al.	
		6996616	Feb 7, 2006	Leighton, F, T., et al.	
		7003572	Feb 21, 2006	Lownsbrough, D. L., et al.	
		7007089	Feb 28, 2006	Freedman, A. T.	
		7010578	Mar 7, 2006	Lewin, D. M., et al.	
		7012900	Mar 14, 2006	Riddle, G.	
		7013342	Mar 14, 2006	Riddle, Guy	
		7032072	Apr 18, 2006	Quinn, Michael J., et al.	
		7039633	May 2, 2006	Dey, J. K., et al.	
		7047300	May 16, 2006	Oehrke, T. L., et al.	
		7054902	May 30, 2006	Toporek, Jerome D., et al.	
		7054935	May 30, 2006	Farber, David A., et al.	
		7058706	Jun 6, 2006	Iyer, Shankar et al.	
		7069177	Jun 27, 2006	Carley, Stephen	
		7096266	Aug 22, 2006	Lewin, Daniel M., et al.	
		7103564	Sep 5, 2006	Ehnebuske	
		7110984	Sep 19, 2006	Spagna	
		7117259	Oct 3, 2006	Rohwer	
		7188085	Mar 6, 2007	Pelletier	
		7206748	Apr 17, 2007	Gruse	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No	Foreign Patent Document	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	T²
		JP-05162529	Jun 29, 1993	Hiroshi, Saito		
		EP-0649121	Apr 19, 1995	IBM		
		EP-0651554	May 3, 1995	Eastman Kodak Co.		
		JP-07066829	Mar 10, 1995	Hiroshi, Saito		
		EP-0801487A2	Oct 15, 1997	Douglas, B. C.		

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. ¹ Applicant's unique citation designation number (optional) ² Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	13	of	21	Attorney Docket No: 0137-US-U1

		EP-0824236A2	Feb 18, 1998	Baird, R. et al.		
		JP-08328583	Dec 13, 1996	NT&T		
		EP-0865180A2	Sep 16, 1998	Chung, P. Y., et al.		
		JP-10027148A2	Jan 27, 1998	ONO, S. et al.		
		JP-10093552	Apr 10, 1998	MURAKAMI, H. et al.		
		EP-1063831A2	Dec 27, 2000	Ogino, T.		
		EP-1104555	Jun 6, 2001	IBM		
		CN-1264476A	Aug 23, 2000	Kenner, Brian et al.		
		IL-140935	Mar 19, 2006	IBM		
		WO-1996/042041A2	Dec 27, 1996	LEVERGOOD, T. M., et al.		
		WO-1997/011429A1	Mar 27, 1997	GRABER, T. E., et al.		
		WO-1997/029423A1	Aug 14, 1997	STANFORD, C. et al.		
		WO-199742582	Nov 13, 1997	Netcast Communications Corp		
		WO-1998/004985A1	Feb 5, 1998	KRIEGSMAN, M.		
		WO-1998/006033A1	Feb 12, 1998	AGRANAT, I. D., et al.		
		WO-199859486	Dec 30, 1998	Telecom PTT		
		WO-199917227	Apr 8, 1999	IBM		
		WO-199948246	Sep 23, 1999	Reuters America Inc		
		WO-199953422	Oct 21, 1999	Inktomi Corp		
		WO-2000/029990A1	May 25, 2000	LEWIS, K. T., et al.		
		WO-2000060861	Oct 12, 2000	Armstrong, J. B.		
		JP-2000207270	Jul 28, 2000	Tomoya, Anzai et al.		
		KR-20010023599	Mar 26, 2001	Malkin et al.		
		JP-2001007844	Jan 12, 2001	Tsukasa, Ogino		
		JP-2001290787	Oct 19, 2001	Hidenori, Araki et al.		
		JP-2001312434	Nov 9, 2001	Toru, Okugawa		
		WO-200193533	Dec 6, 2001	Aerocast Com Inc		
		JP-2002522995	Jul 23, 2002	IBM		
		JP-2005124165	May 12, 2005	IBM		
		CA-2202572	Oct 14, 1998	NANDY, B. et al.		
		GB-2281793A1	Mar 15, 1995	GRAHAM, D. W., et al.		

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	14	of	21	Attorney Docket No: 0137-US-U1

		CA-2288488	Jun 16, 2000	Cohen, Ariel et al.		
		CA-2335661	Mar 9, 2001	IBM		
		CA-2335662	Mar 9, 2001	IBM		
		GB-2353877	Apr 28, 2004	IBM		
		CA-2467998	Apr 25, 2006	IBM		
		JP-3566626	Jun 18, 2004	IBM		
		JP-3762649	Jan 20, 2006	IBM		
		AU-763380	Jun 11, 2003	IBM		
		TW-NI176482	Aug 19, 2003	IBM		
		CN-ZL99810853.7	Aug 12, 1999	IBM		

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		"Exporting Web Server Final Report", http://www.cs.technion.ac.il/Labs/Lccn/projects/spring97/project4/final_report.html (downloaded 07/07/07). Spring 1997	
		"IBM Technical Disclosure Bulletin; Local Area Network Server Replacement Procedure", Vol. 38, No.1 (January 1995) , 235-236	
		JP 10(1998) - 27148 machine translation prepared by Japanese Patent Office (JPO), published Jan-27-98, Title: "Server System for Internet," Applicant: Hitachi , 23 pgs.	
		Patent Abstracts of Japan "Server System for Internet", Pub. No. 10-027148, pub. date Jan-27-98, Applicant: Hitachi, computer translation , 12 pgs.	
		Patent Abstracts of Japan, "Electronic Mail Multiplexing System and Communication Control Method in the System", (Appln. No. JP19930162529), (06/30/1993) (Pub. No. JP 7066829)	
		Patent Abstracts of Japan, "Method and Device for Repeating and Converting Information", (Appln. No. JP19960328583) (Pub. No. JP10171727) 06-26-1998	
		ADLER, R. M., "Distributed Coordination Models for Client/Server Computing", Computer 28, 4 April 1995 , 14-22	
		AGGARWAL, A. et al., "Performance of Dynamic Replication Schemes for an Internet Hosting Service", Technical Report, AT&T Labs October 1998	
		ALMEROTH, K. et al., "Scalable Delivery of Web Pages Using Cyclic Best-Effort (UDP) Multicast", IEEE INFOCOM San Francisco, California, USA June 1998 , pp. 1-24	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Sheet	15	of	21

Complete if Known

Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		ANDRESEN, D. et al., "Multiprocessor scheduling with client resources to improve the response time of WWW applications", ACM Press, NY, Proc. 11th Inti Conf. on Supercomputing (Austria, ICS '97) July 1997 , 92-99	
		ANDRESEN, et al., "SWEB: Towards a Scalable World Wide Web Server on Multicomputers", Proc. IPPS 4/15/1996 , 850-856	
		AWERBUCH, et al., "Competitive Distributed File Allocation", in Proc. of the 25th Ann. ACM Symp. on Theory of Computing May 1993 , pp. 164-173	
		AWERBUCH, B. et al., "Distributed Paging for General Networks", In Proc. of the 7th ACM-SIAM Symposium on Discrete Algorithms, January 1996 , pp. 574-583	
		BAENTSCH, M et al., "Enhancing the Web's Infrastructure: From Caching to Replication", IEEE Internet Computing, Mar. 1997 , 1(2):18-27	
		BAENTSCH, M. et al., "Introducing Application-Level Replication and Naming into Today's Web", Computer Networks and ISDN Systems Vol. 28, Nos. 7-11 May 1996 , pp. 921-930	
		BARTAL, Y. et al., "Competitive Algorithms for Distributed Data Management", 24th Annual ACM STOC, 5/92, Victoria, B.C. Canada	
		BASTURK, E. et al., "Using network layer anycast for load distribution in the Internet", Tech. Rep., IBM TJ. Watson Research Center July 1997 , 21 pgs	
		BERNERS-LEE, T. et al., RFC 1738--Uniform Resource Locators Dec 1994 , pp. 1-25	
		BERNERS-LEE, T. et al., "Hypertext Markup Language - 2.0", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1866 Nov 1995 , pp. 1-77	
		BERNERS-LEE, T. et al., "Hypertext Transfer Protocol -- HTTP/1.0", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1945 May 1996 , pp. 1-60	
		BERNERS-LEE, T. et al., "Hypertext Transfer Protocol -- HTTP/1.0", Internet Engineering Task Force (IETF), Internet Draft, draft-ietf-http-v10-spec-00.ps March 1995 , pp. 1-59	
		BERRA, P. B. et al., "Architecture for Distributed Database Systems", Computer Communications vol. 13, No. 4 May 1, 1990 , pp. 217-231	
		BESTAVROS, A., "Demand-Based Document Dissemination to Reduce Traffic and Balance Load in Distributed Information Systems", In Proc. IEEE Symp. on Parallel and Distributed Processing, San Antonio, TX Oct. 1995 , 8 pgs.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	16	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		BESTAVROS, et al., "Server-Initiated Document Dissemination for the WWW", IEEE Data Engineering Bulletin Sept. 1996 , 19(3): 3-11	
		BESTAVROS, A., "Speculative Data Dissermination and Service to Reduce Server Load Network Traffic and Service Time in Distributed Information Systems", In Proc. ICDE '96: The 1996 Int'l Conf. on Data Enqineering (Mar. 1996) , 4 Pages	
		BHATTACHARJEE, S. et al., "Application-Layer Anycasting", in Proc. IEEE INFOCOM '97 April 1997 , 9 pgs.	
		BRAUN, H. et al., "Web traffic characterization: an assessment of the impact of caching documents from NCSA's web server", Comput. Netw. ISDN Syst. 28, 1-2 Dec. 1995 , 37-51	
		BRISCO, T., "DNS Support for Load Balancing", IETF RFC 1794 April 1995 , pp. 1-7	
		CARTER, et al., "Dynamic server selection using bandwidth probing in wide-area networks", Tech. Rep. BU-CS-96-007, Compo Sci. Dept., Boston University Mar. 1996 , 1-20	
		CARTER, et al., "Server selection using dynamic path characterization in Wide-Area Networks", IEEE INFOCOM '97 1997 , pp. 1014-1021	
		CARTER, J. L. et al., "Universal Classes of Hash Functions", Journal of Computer and System Sciences Vol. 18, No.2 April 1979 , 106-112	
		CATE, V., "Alex: a global file system", in Proc. Usenix Conf. on File Systems May 1992 , pp. 1-11	
		CHANKHUNTHOD, A. et al., "A Hierarchical Internet Object Cache", Proc. Of the 1996 USENIX Technical Conf. Jan. 1996 , 153-163	
		COHEN, J. et al., "Cache Array Routing Protocol v1.1", http://tools.ietf.org/id/draft-vinod-carp-v1-01.txt (Last-Modified: Wed, 01 Oct 1997) 29 Sept. 1997 , 8 pages	
		COLAJANNI, et al., "Adaptive TTL schemes for load balancing of distributed Web servers", SIGMETRICS Perform. Eval. Rev. 25,2 Sep. 1997 , 36-42	
		COLAJANNI, M. et al., "Scheduling Algorithms for Distributed Web Servers", Intl Conf. on Distributed Computing Systems (ICDCS), Baltimore, Maryland, USA May 1997 , pp. 169-176	
		CROVELLA, et al., "Dynamic server selection in the Internet", 3rd IEEE Workshop on the Arch. and Implementation of High Performance Computer Sys. '95 Aug. 1995 , pp. 158-162	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	17	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		DANZIG, P. B. et al., "An analysis of wide-area name server traffic: a study of the Internet Domain Name System", Conf. Proc. Communications Architectures & Protocols; D. Oran, Ed. SIGCOMM '92; ACM Press, New York, NY Aug. 1992 , 281-292	
		DE BRA, P.M.E. et al., "Information Retrieval in the World Wide Web: Making Client-Based Searching Feasible", Computer Networks and ISDN System, NL, North Holland Publishing, Amsterdam, Vol. 27, No.2, ISSN: 0169-7552 11/01/1994 , 183-192	
		DEERING, S. E. et al., "Multicast routing in datagram internetworks and extended LANs", ACM Trans.Comput. Syst. 8,2 May. 1990 , 85-110	
		DOI, K., "Super Proxy Script -- How to make distributed proxy servers by URL hashing", Sharp Corp., http://naragw.sharp.co.jp/sps/; download 07/07/07. dates unknown (1996-2000)	
		ERIKSSON, H., "MBONE: The Multicast Backbone", Communications of the ACM vol. 37, no. 8 Aug 1994 , pp. 54-60	
		FEELEY, M. et al., "Implementing Global Memory Management in a Workstation Cluster", In Proc. 15th ACM Symp. on Operating Systems Principles Dec.1995 , 201-212	
		FIELDING, R. et al., "Hypertext Transfer Protocol -- HTTP/1.1", Internet Engineering Task Force (IETF) Internet Draft, draft-ietf-http-v11-spec-00.txt Nov 1995 , pp. 1-81	
		FIELDING, R. et al., "Hypertext Transfer Protocol -- HTTP/1.1", Internet Engineering Task Force (IETF) Request for Comments (RFC) 2068 Jan 1997 , pp. 1-162	
		FLOYD, S. et al., "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing", In Proc. of ACM SIGCOMM '95, August, 1995 , 342-356	
		FOX, A. et al., "A Framework for Separating Server Scalability and Availability from Internet Application Functionality", PhD thesis, University of California, Berkeley, 1998 , 163 pgs	
		FOX, A. et al., "Cluster-based scalable network services", Proc. 16th ACM Symp. on Operating Systems Principles (Saint Malo, France, Oct. 05 - 08, 1997), W. M. Waite, Ed. SOSP '97. ACM Press, New York, NY , 78-91	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	18	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		GADDE, S. et al., "Reduce, reuse, recycle: An approach to building large internet caches", in Workshop on Hot Topics in Operating Systems (HotOS) April 1997 , pp. 93-98	
		GOLDSZMIDT, M. et al., "Load Distribution for Scalable Web Servers: Summer Olympics 1996--A Case Study", In Proc. 8th IFIPIIEEE Int'l Workshop on Distributed Systems: Operations and Management , Sydney, Australia. Oct. 1997 , 10 pgs.	
		GRIGNI, M. et al., "Tight Bounds on Minimum Broadcasts Networks", SIAM J. Disc. Math. 4 (May, 1991) , 207-222	
		GUYTON, et al., "Locating nearby copies of replicated Internet servers", Proc. ACM SIGCOMM '95; pp. 288-298 October 1995	
		GWERTZMAN, J. et al., "The Case for Geographical Push-Caching", Proc. Workshop on Hot OS '95 (05/04/1995) , 51-55	
		GWERTZMAN, J. et al., "World-Wide Web Cache Consistency", Proc. 1996 USENIX Tech. Conf., pgs.141-151, San Diego, CA Jan. 1996,	
		JEFFERY, C. et al., "Proxy sharing proxy servers", In Proc. IEEE etaCOM Conf. May 1996 , pp. 116-119	
		KARGER, D. et al., "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web", In Proc. 29th Annual ACM Symp. on Theory of Computing May 1997 , 654-663	
		KIM, Y. J. et al., "Clustered multi-media NOD: Popularity-based article prefetching and placement", Sixteenth IEEE Symp. on Mass Storage Systems, San Diego, CA (March 15-18,1999 , pp. 194-202	
		KORKEA-AHO, M., "Scalability in Distributed Multimedia Systems", Technical report TKO-B128, Helsinki University of Technology	
		KRISHNAMURTHY, B. et al., "Study of piggyback cache validation for proxy caches in the World Wide Web", in: Symp. on Internet Technology and Systems, USENIX Association Dec. 1997	
		KROEGER, T. M. et al., "Exploring the Bounds of Web Latency Reduction from Caching and Prefetching", Proc. Usenix Symp. Internet Technologies and Systems, Usenix December 1997 , pp. 13-22	
		KWAN, et al., "NCSA's World Wide Web Server: Design and Performance", IEEE Nov. 1995 , pp. 68-74	

EXAMINER	/Thomas Dailey/	**DATE CONSIDERED**	10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	19	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		LITTLE, T.D. C. et al., "Selection and Dissemination of Digital Video via the Virtual Video Browser", Multimedia Tools and Applications vol. 1, No. 2 (Netherlands) Jun. 1995 , pp. 149-172	
		LUOTONEN, et al., "World-Wide Web Proxies", CERN modified May 24,1994 Apr. 1994 , 1-8	
		MALKIN, G., "RIP Version 2 Carrying Additional Information", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1388 Jan 1993 , pp. 1-7	
		MALPANI, R. et al., "Making World Wide Web Caching Servers Cooperate", In Proc. 4th Int'l. World Wide Web Conf. (downloaded from http://www.w3.org/ConferencesIWWW4/Papers/59/ on 07/07/07) Dec. 1995 , 10 pages	
		MOCKAPETRIS, P., "Domain Names - Concepts and Facilities", IETF RFC 1034 Nov 1987 , 55 pgs.	
		MOCKAPETRIS, P., "Domain Names - Implementation and Specification", IETF RFC 1035 Nov 1987 , 55 pgs.	
		MOURAD, et al., "Scalable Web Server Architectures", iscc, 2nd IEEE Symposium on Computers and Communications (ISCC '97) July 1997 , pp. 12-16	
		MOY, J., "OSPF Version 2", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1583 March 1994 , 132 pgs.	
		NIKI, HITOSHI , "Storage Network", Computopia Vol. 36, No. 415, Japan, Computer Age Co., Ltd. (translation) April 1, 2001 , P. 52	
		OGUCHI, et al., "A Study of Caching Proxy Mechanisms Realized on Wide Area Distributed Networks", High Performance Distributed Computing, 5th Int'l Symposium August 1996 , pp. 443-449	
		PALMER, M. et al., "Fido: A Cache that Learns to Fetch", In Proc. the 17th Int'l Conf. on Very Large Data Bases Sept. 1991 , 255-264	
		PANIGRAHY, R., "Relieving Hot Spots on the World Wide Web", Master's thesis, MIT EECS June 1997 , pp. 1-66	
		PARRIS, C. et al., "A Dynamic Connection Management Scheme for Guaranteed Performance Services in Packet-Switching Integrated Services Networks", UC Berkeley Computer Science Division Tech. Report TR-93-005 1993 , 37 pgs.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	20	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		PARRIS, C. et al., "The Dynamic Management of Guaranteed Performance Connections in Packet Switched Integrated Service Networks", UC Berkeley Computer Science Division and International Computer Science Institute Tech. Report CSD-94-859 1994	
		PARTRIDGE, C. et al., "Host Anycasting Service", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1546 Nov 1993 , pp. 1-9	
		PETRI, S. et al., "Load Balancing and Fault Tolerance in Workstation Clusters. Migrating Groups of Communicating Processes", Operating Systems Review, Vol. 29, No.4 Oct. 1995 , 25-36	
		PLAXTON, G. C. et al., "Fast Fault-Tolerant Concurrent Access to Shared Objects", In Proc. 37th IEEE Symp. of Foundations of Computer Science Oct 1996 , pp. 570-579	
		POVEY, D. et al., "A distributed internet cache", in Proc. of the 20th Australasian Computer Science Conf. Feb. 1997	
		RABIN, M. O., "Efficient dispersal of information for security, load balancing, and fault tolerance", J.ACM 36, 2 (Apr. 1989) , pp. 335-348	
		RABINOVICH, M. et al., "Dynamic Replication on the Internet Work Project No. 3116-17-7006", AT&T Labs Research Mar. 5, 1998	
		RABINOVICH, M. et al., "RaDaR: A Scalable Architecture for a Global Web Hosting Service", WWW8 May 1999	
		REISSLEIN, M. et al., "Interactive video streaming with proxy servers", in Proc. of First International Workshop on Intelligent Multimedia Computing and Networking (IMMCN) 2000 , pp. 1-16	
		RODRIGUEZ, P. et al., "Improving the WWW: Caching or Multicast?", Computer Networks and ISDN Systems North Holland Publishing, Amsterdam, NL, vol. 30, No. 22-23, ISSN: 0169-7552 Nov. 1998 , pp. 2223-2243	
		ROSS, K. W., "Hash-Routing for Collections of Shared Web Caches", IEEE Network Magazine 11, 7:37-44 Nov-Dec 1997 , pp. 1-21	
		SEN, S. et al., "Proxy Prefix Caching for Multimedia Streams", IEEE June 1999 , 10 pgs.	
		SMITH, NEIL , "What can Archives offer the World Wide Web?", Technical Report 11, University of Kent, Computing Laboratory, University of Kent, Canterbury, UK March 1994 , 1-12	
		TAKAHASHI, TAKAO , "How to customize Apache Q&A -- Web server using Apache", AI Publishing, April 30, 1999, First Edition , pp. 63-64 (translation)	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	21	of	21

Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		THALER, D. G. et al., "Using name-based mappings to increase hit rates", IEEE/ACM Trans. Netw. 6,1 Feb. 1998 , 1-14	
		TOPOLCIC, C., "Experimental Internet Stream Protocol, Version 2 (ST-II)", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1190 Oct 1990 , pp. 1-148	
		TOUCH, J. et al., "The X-Bone", Third Global Internet Mini-Conference at Globecom '98, Sydney, Australia (pp. 44-52 of mini-conference) Nov 1998 , pp. 59-68	
		TRAINA, P., "BGP-4 Protocol Analysis", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1774 March 1995 , pp. 1-10	
		VIN, H., "Multimedia Broadcasting Over the Internet: Part 1", IEEE Multimedia IEEE Computer Society, US, vol. 5, NR. 4 Oct. 1998 , pp. 78-82	
		VITTER, J. S. et al., "Optimal Prefetching via Data Compression", Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science October 1991 , 21 pages	
		WESSELS, D., "Configuring Hierarchical Squid Caches", Aug. 19 1997 , 62 pgs.	
		WESSELS, DUANE , "Intelligent Caching for World-Wide Web Objects", Masters Thesis, University of Colorado (also presented at INET '95 in June '95) Jan 1995 , 1-85	
		WESSELS, D. et al., "Internet Cache Protocol (ICP), Version 2", Internet Engineering Task Force (IETF) Request for Comments (RFC) 2186 Sept 1997 , pp. 1-9	
		WOLFSON, O. et al., "An Adaptive Data Replication Algorith", ACM Transactions on Database Systems (TODS) Vol. 22(4) June 1997 , pp. 255-314	
		XIAO, LI et al., "Exploiting neglected data locality in browsers", in Proc. 10th Int'l World Wide Web Conference, (WWW10), Hong Kong (extended abstract) May 1-5 2001 , 2 pgs.	

EXAMINER /Thomas Dailey/ **DATE CONSIDERED** 10/31/2014

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

ALL REFERENCES CONSIDERED EXCEPT WHERE LINED THROUGH. /T.D

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12408681	FULLAGAR ET AL.
	Examiner	Art Unit
	NEERAJ UTREJA	2452

CPC- SEARCHED		
Symbol	Date	Examiner

CPC COMBINATION SETS - SEARCHED		
Symbol	Date	Examiner

US CLASSIFICATION SEARCHED			
Class	Subclass	Date	Examiner
709	219, 223, 226	10/31/2014	TJD

SEARCH NOTES		
Search Notes	Date	Examiner
EAST text search	3/23/2011	NU
Google Patents search	3/23/2011	NU
double patenting / inventor search	3/23/2011	NU
EAST text search updated	11/10/2011	NU
double patenting / inventor search	11/10/2011	NU
Google Patents search: "long tail popular cdn"	11/10/2011	NU
Keyword searching in EAST for amended subject matter	7/5/2013	TJD
Updated search	9/26/2014	TJD
NPL search utilizing Google	9/26/2014	TJD
Updated search	10/31/2014	TJD
Reviewed submitted IDSs	10/31/2014	TJD

INTERFERENCE SEARCH			
US Class/ CPC Symbol	US Subclass / CPC Group	Date	Examiner
709	219,223,226	10/31/2014	TJD

/NEERAJ UTREJA/ Examiner.Art Unit 2452	

UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

BIB DATA SHEET

CONFIRMATION NO. 9965

SERIAL NUMBER	FILING or 371(c) DATE RULE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
12/408,681	03/21/2009	709	2452	0137-US-U1

APPLICANTS

INVENTORS

David Fullagar, Boulder, CO;
Christopher Newton, Thousand Oaks, CA;
Laurence R. Lipstone, Calabasas, CA;

**** CONTINUING DATA ***************************

This appln claims benefit of 61/042,412 04/04/2008

**** FOREIGN APPLICATIONS ***************************

**** IF REQUIRED, FOREIGN FILING LICENSE GRANTED ****

04/01/2009

Foreign Priority claimed ☐ Yes ☑ No 35 USC 119(a-d) conditions met ☐ Yes ☑ No Verified and Acknowledged /NEERAJ K UTREJA/ Examiner's Signature	☐ Met after Allowance Initials	STATE OR COUNTRY	SHEETS DRAWINGS	TOTAL CLAIMS	INDEPENDENT CLAIMS
		CO	6	12	4

ADDRESS

LEVEL 3 COMMUNICATIONS, LLC
Attn: Patent Docketing
1025 Eldorado Blvd.
Broomfield, CO 80021
UNITED STATES

TITLE

HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)

FILING FEE RECEIVED	FEES: Authority has been given in Paper	☐ All Fees
1752	No._____ to charge/credit DEPOSIT ACCOUNT	☐ 1.16 Fees (Filing)
	No._____ for following:	☐ 1.17 Fees (Processing Ext. of time)
		☐ 1.18 Fees (Issue)
		☐ Other _____
		☐ Credit

BIB (Rev. 05/07).

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12408681	FULLAGAR ET AL.
	Examiner	**Art Unit**
	THOMAS DAILEY	2452

CPC						
Symbol					**Type**	**Version**
H04N		7	/	17354	F	2013-01-01
H04N		21	/	222	I	2013-01-01
H04N		21	/	23106	I	2013-01-01
H04N		21	/	25891	I	2013-01-01
H04N		21	/	472	I	2013-01-01
H04L		67	/	2814	I	2013-01-01
H04L		67	/	2842	I	2013-01-01
			/			
			/			
			/			
			/			
			/			
			/			
			/			
			/			

CPC Combination Sets								
Symbol					**Type**	**Set**	**Ranking**	**Version**
			/					
			/					

NONE		**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	29	
/THOMAS DAILEY/ Primary Examiner.Art Unit 2452	10/31/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12408681	FULLAGAR ET AL.
	Examiner	**Art Unit**
	THOMAS DAILEY	2452

US ORIGINAL CLASSIFICATION		INTERNATIONAL CLASSIFICATION												
CLASS	SUBCLASS	CLAIMED					NON-CLAIMED							
709	226	G	0	6	F	15 / 173 (2006.01.01)								
CROSS REFERENCE(S)														
CLASS	SUBCLASS (ONE SUBCLASS PER BLOCK)													
709	219	223												

NONE		Total Claims Allowed:	
(Assistant Examiner)	(Date)	29	
/THOMAS DAILEY/ Primary Examiner.Art Unit 2452	10/31/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

Issue Classification	Application/Control No.	Applicant(s)/Patent Under Reexamination
	12408681	FULLAGAR ET AL.
	Examiner	**Art Unit**
	THOMAS DAILEY	2452

☐ Claims renumbered in the same order as presented by applicant ☐ CPA ☐ T.D. ☐ R.1.47

Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original	Final	Original
1	1	17	17												
2	2	18	18												
3	3	19	19												
4	4	20	20												
5	5	21	21												
6	6	22	22												
7	7	23	23												
8	8	24	24												
9	9	25	25												
10	10	26	26												
11	11	27	27												
12	12	28	28												
13	13	29	29												
14	14														
15	15														
16	16														

NONE		**Total Claims Allowed:**	
(Assistant Examiner)	(Date)	29	
/THOMAS DAILEY/ Primary Examiner.Art Unit 2452	10/31/2014	O.G. Print Claim(s)	O.G. Print Figure
(Primary Examiner)	(Date)	1	1

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	34	("5913028" "5913033" "5918010" "5919247" "5920701" "5931904" "5933832" "5933835" "5935207" "5940831" "5944780" "5951694" "5956489" "5956716" "5958008" "5961596" "5966440" "5968121" "5973696" "5978791" "5983214" "5983227" "5987606" "5991809" "5996025" "6002720" "6003030" "6012090" "6014686" "6014698" "6016509" "6016512" "6018516" "6021426").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:00
L2	33	("20010029525" 20010049732A1 "20010051980" "20010056500" "20020018449" "20020023164" "20020023165" "20020042817" "20020046273" "20020046405" "20020049857" "20020059592" "20020066038" "20020073199" "20020078233" "20020082999" "20020083124" "20020089470" "20020092026" "20020099850" "20020124080" "20020129134" "20020131645" "20020143798" "20020143888" "20020147774" "20020152318" "20020163882" "20020199016" "20030009444" "20030018966" "20030028623" "20030028626" "20030028777").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:01
L3	34	("20010029525" "20010049732" "20010051980" "20010056500" "20020018449" "20020023164" "20020023165" "20020042817" "20020046273" "20020046405" "20020049857" "20020059592" "20020066038" "20020073199" "20020078233" "20020082999" "20020083124" "20020089470" "20020092026" "20020099850" "20020124080" "20020129134" "20020131645" "20020143798" "20020143888" "20020147774" "20020152318" "20020163882" "20020199016" "20030009444" "20030018966" "20030028623" "20030028626" "20030028777").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:01
L4	34	("20030055972" "20030061263" "20030061280" "20030065703" "20030065762" "20030078888" "20030078889" "20030079027" "20030095660" "20030105604" "20030154239" "20030158923" "20030158928" "20040022194" "20040139097" "20040177148" "20050033858" "20050038851"	US-PGPUB; USPAT	OR	ON	2014/10/31 14:01

		"20050100027" "20050114296" "20050262104" "20060143293" "4495570" "4591983" "4594704" "4726017" "4803641" "4839798" "4847784" "4920432" "4922417" "4943932" "4949187" "4949248").pn.				
L5	34	("5029232" "5130792" "5132992" "5136716" "5172413" "5191573" "5253275" "5253341" "5287499" "5287537" "5291554" "5341477" "5371532" "5410343" "5414455" "5442389" "5442390" "5442749" "5471622" "5475615" "5508732" "5515511" "5519435" "5528281" "5539621" "5542087" "5544313" "5544327" "5550577" "5550863" "5550982" "5557317" "5568181" "5572643").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:02
L6	34	("5590288" "5592611" "5594910" "5603026" "5614940" "5619648" "5623656" "5625781" "5627829" "5630067" "5633999" "5634006" "5638443" "5644714" "5646676" "5649186" "5659729" "5666362" "5671279" "5675734" "5682512" "5699513" "5712979" "5715453" "5721914" "5734719" "5734831" "5740423" "5742762" "5751961" "5761507" "5761663" "5764906" "5774660").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:02
L7	34	("5774668" "5777989" "5778187" "5781909" "5784058" "5796952" "5799141" "5802106" "5802291" "5812769" "5815664" "5819092" "5828847" "5832069" "5832506" "5832514" "5835718" "5845090" "5845303" "5856974" "5862339" "5867706" "5867799" "5870546" "5870559" "5878212" "5884038" "5890171" "5893116" "5894554" "5896533" "5898456" "5903723" "5907704").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:03
L8	34	("6023470" "6029175" "6029176" "6035332" "6038216" "6038310" "6041324" "6044405" "6046980" "6049831" "6052718" "6052730" "6065051" "6065062" "6070191" "6078943" "6081829" "6081835" "6081840" "6092112" "6092204" "6094706" "6098078" "6098096" "6105028" "6108673" "6108703" "6112231" "6112239" "6112240" "6115357" "6115752" "6119143" "6125388").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:03
L9	34	("6125394" "6128601" "6128623" "6128660" "6130890" "6131095" "6134583" "6144375" "6144702" "6151624" "6154744" "6154753" "6161137" "6163779" "6173311" "6173322" "6178160" "6181867" "6185598" "6185619" "6189030" "6189039" "6195680" "6205120" "6226618" "6226642" "6230196" "6233623" "6240462" "6249810" "6256675" "6263313" "6266335" "6266699").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:03
L10	34	("6269394" "6272566" "6275470" "6282569" "6286045" "6298041" "6314565" "6317787" "6324580" "6324582" "6330602" "6332157" "6338044" "6343298" "6345294" "6347085" "6351775" "6351776" "6360256" "6370571" "6370580" "6389462" "6398245" "6405252" "6405257" "6412000" "6415280" "6415368" "6418421" "6421714" "6421726" "6427170" "6430618" "6442588").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:04
L11	34	("6453319" "6456630" "6460082" "6460085" "6463508" "6466949" "6470389" "6473405" "6480893" "6484143" "6484204" "6487555"	US-PGPUB; USPAT	OR	ON	2014/10/31 14:04

		"6490580" "6493707" "6496856" "6502125" "6502205" "6502215" "6505248" "6529477" "6553413" "6553420" "6557054" "6564251" "6577595" "6581090" "6584083" "6587837" "6591299" "6611862" "6625643" "6651141" "6654344" "6654807").pn.				
L12	34	("6658464" "6665706" "6678659" "6691148" "6694358" "6699418" "6708137" "6718328" "6741563" "6751673" "6754699" "6754706" "6763377" "6763388" "6778502" "6799221" "6801576" "6834306" "6842604" "6859791" "6870851" "6874032" "6901604" "6915307" "6915329" "6928442" "6934255" "6950623" "6963910" "6963980" "6963981" "6970432" "6973490" "6981050").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:04
L13	20	("6981180" "6996616" "7003572" "7007089" "7010578" "7012900" "7013342" "7032072" "7039633" "7047300" "7054902" "7054935" "7058706" "7069177" "7096266" "7103564" "7110984" "7117259" "7188085" "7206748").pn.	US-PGPUB; USPAT	OR	ON	2014/10/31 14:05
S1	60	("20040205162" | "20010027491" | "7103645" | "20050060493" | "20030140111" | "6212565" | "20080147866" | "6859840" | "7274658" | "20100312861" | "20050240966" | "7219153" | "20020016835" | "20030149581" | "20060015574" | "20080065724" | "6591288" | "6594260" | "6839758" | "6999988" | "7254645" | "20060112176" | "20080071859" | "20020040404" | "20110145386" | "7562153" | "20040064832" | "20040162871" | "20030031176" | "20050021863" | "7024466" | "7185052" | "6157648" | "20100257258" | "20070055764" | "20020198953" | "6094680" | "6463454" | "6785704" | "7373644" | "20020010798" | "20020174168" | "20110099290" | "20020016831" | "20020087797" | "20020116444" | "20020133537" | "20030112792" | "6085193" | "6598121" | "6732237" | "6976090" | "7577754" | "20040010588" | "20020184357" | "20050010653" | "5777988" | "5815662" | "6741990" | "20100169772" | "7024466").PN.	US-PGPUB; USPAT	ADJ	ON	2013/07/05 11:23
S2	1371	popular$5 with (check$6 or determin$6) same (server)	US-PGPUB; USPAT	ADJ	ON	2013/07/05 14:30
S3	240	popular$5 with (check$6 or determin$6) with (content or file) with (server)	US-PGPUB; USPAT	ADJ	ON	2013/07/05 14:30
S4	0	S3 and ("709" or "725" or "370").ccls.	US-PGPUB; USPAT	ADJ	ON	2013/07/05 14:31
S5	162	S3 and ("709" or "725" or "370").clas.	US-PGPUB; USPAT	ADJ	ON	2013/07/05 14:31
S6	78	popular$5 with (option$7) with (content or file) with (server)	US-PGPUB; USPAT	ADJ	ON	2013/07/05 14:53
S7	60	S6 and ("709" or "725" or "370").clas.	US-	ADJ	ON	2013/07/05

			PGPUB; USPAT			14:53
S8	18	S6 not S7	US-PGPUB; USPAT	ADJ	ON	2013/07/05 14:58
S9	164	popular$5 with (certain or particul$5) with (content or file) with (server)	US-PGPUB; USPAT	ADJ	ON	2013/07/05 15:04
S10	114	S9 and ("709" or "725" or "370").clas.	US-PGPUB; USPAT	ADJ	ON	2013/07/05 15:05
S11	1	"20070233820"	US-PGPUB; USPAT	ADJ	ON	2013/07/05 16:33
S12	214	popular$5 with (certain or particul$5) with (content or file) with (server)	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:24
S13	887	(CDN or content delivery network) same (popula$6)	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:26
S14	115	(CDN or content delivery network) same (popula$6) with deter$6	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:26
S15	60	("20040205162" | "20010027491" | "7103645" | "20050060493" | "20030140111" | "6212565" | "20080147866" | "6859840" | "7274658" | "20100312861" | "20050240966" | "7219153" | "20020016835" | "20030149581" | "20060015574" | "20080065724" | "6591288" | "6594260" | "6839758" | "6999988" | "7254645" | "20060112176" | "20080071859" | "20020040404" | "20110145386" | "7562153" | "20040064832" | "20040162871" | "20030031176" | "20050021863" | "7024466" | "7185052" | "6157648" | "20100257258" | "20070055764" | "20020198953" | "6094680" | "6463454" | "6785704" | "7373644" | "20020010798" | "20020174168" | "20110099290" | "20020016831" | "20020087797" | "20020116444" | "20020133537" | "20030112792" | "6085193" | "6598121" | "6732237" | "6976090" | "7577754" | "20040010588" | "20020184357" | "20050010653" | "5777988" | "5815662" | "6741990" | "20100169772" | "7024466").PN.	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S16	1916	popular$5 with (check$6 or determin$6) same (server)	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S17	391	popular$5 with (check$6 or determin$6) with (content or file) with (server)	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S18	0	S17 and ("709" or "725" or "370").ccls.	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S19	271	S17 and ("709" or "725" or "370").clas.	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59

S20	91	popular$5 with (option$7) with (content or file) with (server)	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S21	69	S20 and ("709" or "725" or "370").clas.	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S22	22	S20 not S21	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S23	214	popular$5 with (certain or particul$5) with (content or file) with (server)	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S24	143	S23 and ("709" or "725" or "370").clas.	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S25	1	"20070233820"	US-PGPUB; USPAT	ADJ	ON	2014/09/26 11:59
S26	40288	709/219,223,226.ccls.	US-PGPUB; USPAT	ADJ	ON	2014/09/26 12:47
S27	200	S16 and S26	US-PGPUB; USPAT	ADJ	ON	2014/09/26 12:47

EAST Search History (Interference)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S28	41	((CDN or content delivery network) same (popula$6)).clm.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/09/26 12:46
S29	7	S28 and 709/226.ccls.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/09/26 12:46
S30	6	S28 and 709/219.ccls.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/09/26 12:47
S31	4	S28 and 709/223.ccls.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/09/26 12:47
S32	40366	709/219,223,226.ccls.	US-PGPUB; USPAT; UPAD	ADJ	ON	2014/09/26 12:47

10/31/2014 2:07:05 PM
C:\Users\tdailey\Documents\EAST\Workspaces\12-408681.wsp

PTO/SB/08a (01-08)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number..

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	1	of	1	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS					
Examiner Initial *	**Cite No**	**Document Number**	**Publication Date**	**Name of Patentee or Applicant of Cited Document**	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear

FOREIGN PATENT DOCUMENTS						
Examiner Initials*	**Cite No**	**Foreign Patent Document**	**Publication Date**	**Name of Patentee or Applicant of cited Document**	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	**T²**

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T²**
		Canadian Examination Report, dated September 8, 2014, Application No. 2,720,353, filed March 21, 2009; Dkt. No. 0137-CA-01 , 4 pgs.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

Electronic Acknowledgement Receipt

EFS ID:	20626941
Application Number:	12408681
International Application Number:	
Confirmation Number:	9965
Title of Invention:	HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)
First Named Inventor/Applicant Name:	David Fullagar
Customer Number:	83579
Filer:	David Dewey Wier/Kathleen McInnish
Filer Authorized By:	David Dewey Wier
Attorney Docket Number:	0137-US-U1
Receipt Date:	06-NOV-2014
Filing Date:	21-MAR-2009
Time Stamp:	15:12:49
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	no

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	137USU1_IDS.pdf	37477 559316f56ae9e3ff7339a93f7b83699a1c77b437	no	5

Warnings:	
Information:	

2	Information Disclosure Statement (IDS) Form (SB08)	137USU1_SB08.pdf	28882 fdaa64de0773b008d7565ebe5351c451c1eaf5df	no	1
Warnings:					
Information:					
This is not an USPTO supplied IDS fillable form					
3	Other Reference-Patent/App/Search documents	CA_2720353_ExRpt_8Sep14_137CA01.pdf	513769 6e746fcc4953a30fc713baef6136e34173d16e97	no	4
Warnings:					
Information:					
		Total Files Size (in bytes):	580128		

This Acknowledgement Receipt evidences receipt on the noted date by the USPTO of the indicated documents, characterized by the applicant, and including page counts, where applicable. It serves as evidence of receipt similar to a Post Card, as described in MPEP 503.

New Applications Under 35 U.S.C. 111
If a new application is being filed and the application includes the necessary components for a filing date (see 37 CFR 1.53(b)-(d) and MPEP 506), a Filing Receipt (37 CFR 1.54) will be issued in due course and the date shown on this Acknowledgement Receipt will establish the filing date of the application.

National Stage of an International Application under 35 U.S.C. 371
If a timely submission to enter the national stage of an international application is compliant with the conditions of 35 U.S.C. 371 and other applicable requirements a Form PCT/DO/EO/903 indicating acceptance of the application as a national stage submission under 35 U.S.C. 371 will be issued in addition to the Filing Receipt, in due course.

New International Application Filed with the USPTO as a Receiving Office
If a new international application is being filed and the international application includes the necessary components for an international filing date (see PCT Article 11 and MPEP 1810), a Notification of the International Application Number and of the International Filing Date (Form PCT/RO/105) will be issued in due course, subject to prescriptions concerning national security, and the date shown on this Acknowledgement Receipt will establish the international filing date of the application.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.	:	12/408,681	Confirmation No.	:	9965
First Applicant	:	FULLAGAR et al.	Art Unit	:	2452
Filed	:	March 21, 2009	Examiner	:	Thomas J. Dailey
Title	:	HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)			
Docket No.	:	0137-US-U1			

Submitted Via EFS-Web

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT PURSUANT TO 37 C.F.R. §§ 1.56, 1.97, AND 1.98

Sir:

The Examiner is respectfully requested to consider the patents, patent applications, publications, or other information cataloged on the enclosed Form (e.g., PTO-1449, PTO-892, PTO/SB/08, etc.) during examination of the above-identified patent application. The order of presentation of the references should not be construed as an indication of importance. These references are submitted for the Examiner's consideration and are submitted pursuant to an Applicant's duty of disclosure under 37 C.F.R. § 1.56.

Copies of all of the cited documents:

☐ are enclosed unless otherwise indicated on the enclosed Form PTO/SB/08a.

☐ have been previously furnished to the Office in prior application No. _____, filed _____, which the present application relies upon for an earlier effective filing date under 35 U.S.C. § 120. Therefore, no copies of the cited documents are included herewith pursuant to 37 C.F.R. § 1.98(d); MPEP § 609(III)(A)(2).

☒ are enclosed unless they are U.S. patents or U.S. patent application publications pursuant to 37 C.F.R. § 1.98(a).

I
BASIS

This information disclosure statement is filed pursuant to

☒ 37 C.F.R. § 1.97(b).
This information disclosure statement is filed either (1) within three months of the filing date of a national application other than a continued prosecution application under 37 C.F.R. § 1.53(d); (2) within three months of the date of entry of the national stage as set forth in 37 C.F.R. § 1.491 in an international application; (3) before the mailing date of a first Office action on the merits; (4) concurrently with the filing of a request for continued examination under 37 C.F.R. § 1.114; or (5) before the mailing of a first Office action after the filing of a request for continued examination under 37 C.F.R. § 1.114, whichever event occurs last.

Accordingly, this information disclosure statement requires no fee and no certification.

☐ 37 C.F.R. § 1.97(c).
This information disclosure statement is filed after the period specified in 37 C.F.R. § 1.97(b), but before the mailing date of any of (1) a final action under 37 C.F.R. § 1.113; (2) a notice of allowance under 37 C.F.R. § 1.311; or (3) an action that otherwise closes prosecution in the application.

Accordingly, this information disclosure statement requires either the fee specified in 37 C.F.R. § 1.17(p) for submission of an information disclosure statement under 37 C.F.R. § 1.97(c), or a statement under 37 C.F.R. § 1.97(e).

☐ 37 C.F.R. § 1.97(d).
This information disclosure statement is filed after the period specified in 37 C.F.R. § 1.97(c) since a:

☐ final action under 37 C.F.R. § 1.113

☐ notice of allowance under 37 C.F.R. § 1.311

was mailed to the Applicant on _____.

Accordingly, this information disclosure statement requires the fee set forth in 37 C.F.R. § 1.17(p) to consider an information disclosure statement under 37 C.F.R. § 1.97(d), and a statement under 37 C.F.R. § 1.97(e).

If this statement crosses in the mail with an Office action, or is otherwise not in the indicated category of 37 C.F.R. § 1.97, it is respectfully requested that this statement be treated in the next appropriate category and made of record. To the extent required, please treat this paper as a conditional petition for acceptance of the information disclosure statement.

II
FEES

Any required fee is being paid as indicated below:

☒ No fee is due.

☐ The $180 fee specified in 37 C.F.R. § 1.17(p) for submission of an information disclosure statement under 37 C.F.R. § 1.97(c) or (d) is due.

☐ This fee is paid by credit card.

☐ Please charge the required fee to Deposit Account No. 505063.

☒ The Commissioner is hereby authorized to charge any additional required fees or credit any overpayments associated with this information disclosure statement to the account.

III

CERTIFICATION

Pursuant to 37 C.F.R. § 1.97(e), I state as follows:

☒ No statement is necessary.

☐ Each item of information contained in this information disclosure statement was first cited in a communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of this information disclosure statement. 37 C.F.R. § 1.97(e)(1).

☐ No item of information contained in this information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to my knowledge after making reasonable inquiry, no item of information contained in this information disclosure statement was known to any individual designated in 37 C.F.R. § 1.56(c) more than three months prior to the filing of this information disclosure statement. 37 C.F.R. § 1.97(e)(2).

Pursuant to 37 C.F.R. § 1.704(d), I state as follows:

☐ Each item of information contained in this information disclosure statement was cited in a communication from a foreign patent office in a counterpart application, and this communication was not received by any individual designated in 37 C.F.R. § 1.56(c) more than thirty days prior to the filing of this information disclosure statement.

IV
FOREIGN LANGUAGE REFERENCES

Pursuant to 37 C.F.R. § 1.98(a)(3)(i), any information disclosure statement filed under 37 C.F.R. § 1.97 shall include a concise explanation of the relevance, as it is presently understood by the individual designated in § 1.56(c) most knowledgeable about the content of the information, of each patent, publication, or other information listed that is not in the English language. The concise explanation may be either separate from the specification or incorporated therein. In compliance with the rules, the Applicant states as follows:

☐ No concise explanation is required since all cited references are in English.

☐ A concise explanation of the relevance of one or more foreign language references is attached.

☐ According to the procedural policy adopted by the PTO with regard to disclosure requirements, the Applicant is satisfying the disclosure requirement pursuant to MPEP § 609.04(a)(III) by submitting herewith the following:

- ☐ English language versions of any non-English language documents.
- ☐ English language versions of one or more search reports or other actions by one or more foreign patent offices in one or more counterpart foreign applications, which indicate the degree of relevance found by the foreign office(s).
- ☐ Pursuant to 37 C.F.R. § 1.98(a)(3)(ii), a copy of a translation is not submitted herewith because a written English-language translation of a non-English-language document, or portion thereof, is not within the possession, custody, or control of, or is readily available to any individual designated in § 1.56(c).

The Applicant has not analyzed these documents and, therefore, declines to comment on their relevance at this time.

☐ One or more English language abstracts for the non-English language references are also enclosed.

V
REMARKS

It is respectfully requested that:

(1) The Examiner sign the enclosed Form to evidence that the cited information has been fully considered by the Patent and Trademark Office during the examination of this application; and

(2) The citations for the cited information be printed on any patent that issues from this application.

The submission of this information disclosure statement by the Applicant shall not be construed as a representation that a search has been performed or that no other material information under 37 C.F.R. § 1.56 exists. See 37 C.F.R. § 1.97(g).

The submission of this information disclosure statement by the Applicant shall not be construed as an admission that any information cited or referred to herein is, or is considered to be, material to the patentability of one or more of the pending claims in this application pursuant to 37 C.F.R. § 1.56(b), unless specifically designated by the Applicant as such. See 37 C.F.R. § 1.97(h).

The submission of this information disclosure statement by the Applicant shall not be construed as an admission that any information cited or referred to herein is, or is considered to be, or even qualifies as "prior art" under 35 U.S.C. § 102 with respect to this invention, unless specifically designated by the Applicant as such.

Notwithstanding any statements by the Applicant, the Examiner is urged to form his or her own conclusion regarding the relevance of the cited information.

VI
CONCLUSION

The Examiner is encouraged to contact the undersigned attorney to discuss any questions concerning this information disclosure statement or any of the information cited or referred to herein.

Respectfully submitted,

Date: November 6, 2014

/JCS/

Jonathan C. Siekmann
Registration No. 58,259
USPTO Customer No. 83579
Level 3 Communications, LLC
1025 Eldorado Blvd.
Broomfield, Colorado 80021
Tel: 720-888-2140
Fax: 720-888-5619

PTO/SB/30EFS (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

REQUEST FOR CONTINUED EXAMINATION(RCE)TRANSMITTAL
(Submitted Only via EFS-Web)

Application Number	12/408,681	Filing Date	2009-03-21	Docket Number (if applicable)	0137-US-U1	Art Unit	2452
First Named Inventor	David Fullagar			Examiner Name	Thomas J. DAILEY		

This is a Request for Continued Examination (RCE) under 37 CFR 1.114 of the above-identified application.
Request for Continued Examination (RCE) practice under 37 CFR 1.114 does not apply to any utility or plant application filed prior to June 8 1995, or to any design application. The Instruction Sheet for this form is located at WWW.USPTO.GOV

SUBMISSION REQUIRED UNDER 37 CFR 1.114

Note: If the RCE is proper, any previously filed unentered amendments and amendments enclosed with the RCE will be entered in the orde in which they were filed unless applicant instructs otherwise. If applicant does not wish to have any previously filed unentered amendment(s entered, applicant must request non-entry of such amendment(s).

☐ Previously submitted. If a final Office action is outstanding, any amendments filed after the final Office action may be considered as a submission even if this box is not checked.

- ☐ Consider the arguments in the Appeal Brief or Reply Brief previously filed on _____
- ☐ Other _____

☒ Enclosed

- ☐ Amendment/Reply
- ☒ Information Disclosure Statement (IDS)
- ☐ Affidavit(s)/ Declaration(s)
- ☐ Other _____

MISCELLANEOUS

☐ Suspension of action on the above-identified application is requested under 37 CFR 1.103(c) for a period of months _____
(Period of suspension shall not exceed 3 months; Fee under 37 CFR 1.17(i) required)

☐ Other _____

FEES

☒ **The RCE fee under 37 CFR 1.17(e) is required by 37 CFR 1.114 when the RCE is filed.**
The Director is hereby authorized to charge any underpayment of fees, or credit any overpayments, to Deposit Account No 505063

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

☒ Patent Practitioner Signature
☐ Applicant Signature

PTO/SB/30EFS (07-09)
Approved for use through 07/31/2012. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Signature of Registered U.S. Patent Practitioner			
Signature	/JCS/	Date (YYYY-MM-DD)	2014-10-21
Name	Jonathan C. Siekmann	Registration Number	58259

This collection of information is required by 37 CFR 1.114. The information is required to obtain or retain a benefit by the public which is t file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.11 and 1.14. This collection estimated to take 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Tim will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions fo reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450.

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.

Privacy Act Statement

The Privacy Act of 1974 (P.L. 93-579) requires that you be given certain information in connection with your submission of the attached form related to a patent application or patent. Accordingly, pursuant to the requirements of the Act, please be advised that: (1) the general authority for the collection of this information is 35 U.S.C. 2(b)(2); (2) furnishing of the informatio solicited is voluntary; and (3) the principal purpose for which the information is used by the U.S. Patent and Trademark Office is to process and/or examine your submission related to a patent application or patent. If you do not furnish the requested information, the U.S. Patent and Trademark Office may not be able to process and/or examine your submission, which may result in termination of proceedings or abandonment of the application or expiration of the patent.

The information provided by you in this form will be subject to the following routine uses:

1. The information on this form will be treated confidentially to the extent allowed under the Freedom of Information Act (5 U.S.C. 552) and the Privacy Act (5 U.S.C. 552a). Records from this system of records may be disclosed to the Department of Justice to determine whether the Freedom of Information Act requires disclosure of these records.

2. A record from this system of records may be disclosed, as a routine use, in the course of presenting evidence to a court, magistrate, or administrative tribunal, including disclosures to opposing counsel in the course of settlement negotiations.

3. A record in this system of records may be disclosed, as a routine use, to a Member of Congress submitting a request involving an individual, to whom the record pertains, when the individual has requested assistance from the Member with respect to the subject matter of the record.

4. A record in this system of records may be disclosed, as a routine use, to a contractor of the Agency having need for the information in order to perform a contract. Recipients of information shall be required to comply with the requirements of the Privacy Act of 1974, as amended, pursuant to 5 U.S.C. 552a(m).

5. A record related to an International Application filed under the Patent Cooperation Treaty in this system of records may be disclosed, as a routine use, to the International Bureau of the World Intellectual Property Organization, pursuant to the Patent Cooperation Treaty.

6. A record in this system of records may be disclosed, as a routine use, to another federal agency for purposes of National Security review (35 U.S.C. 181) and for review pursuant to the Atomic Energy Act (42 U.S.C. 218(c)).

7. A record from this system of records may be disclosed, as a routine use, to the Administrator, General Services, or his/her designee, during an inspection of records conducted by GSA as part of that agency's responsibility to recommend improvements in records management practices and programs, under authority of 44 U.S.C. 2904 and 2906. Such disclosure shall be made in accordance with the GSA regulations governing inspection of records for this purpose, and any other relevant (i.e., GSA or Commerce) directive. Such disclosure shall not be used to make determinations about individuals.

8. A record from this system of records may be disclosed, as a routine use, to the public after either publication of the application pursuant to 35 U.S.C. 122(b) or issuance of a patent pursuant to 35 U.S.C. 151. Further, a record ma be disclosed, subject to the limitations of 37 CFR 1.14, as a routine use, to the public if the record was filed in an application which became abandoned or in which the proceedings were terminated and which application is referenced by either a published application, an application open to public inspections or an issued patent.

9. A record from this system of records may be disclosed, as a routine use, to a Federal, State, or local law enforcement agency, if the USPTO becomes aware of a violation or potential violation of law or regulation.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application No.	:	12/408,681	Confirmation No.	:	9965
First Applicant	:	FULLAGAR et al.	Art Unit	:	2452
Filed	:	March 21, 2009	Examiner	:	Thomas J. Dailey
Title	:	HANDLING LONG-TAIL CONTENT IN A CONTENT DELIVERY NETWORK (CDN)			
Docket No.	:	0137-US-U1			

Submitted Via EFS-Web

SUPPLEMENTAL INFORMATION DISCLOSURE STATEMENT PURSUANT TO 37 C.F.R. §§ 1.56, 1.97, AND 1.98

Sir:

The Examiner is respectfully requested to consider the patents, patent applications, publications, or other information cataloged on the enclosed Form (e.g., PTO-1449, PTO-892, PTO/SB/08, etc.) during examination of the above-identified patent application. The order of presentation of the references should not be construed as an indication of importance. These references are submitted for the Examiner's consideration and are submitted pursuant to an Applicant's duty of disclosure under 37 C.F.R. § 1.56.

Copies of all of the cited documents:

☐ are enclosed unless otherwise indicated on the enclosed Form PTO/SB/08a.

☐ have been previously furnished to the Office in prior application No. _____, filed _____, which the present application relies upon for an earlier effective filing date under 35 U.S.C. § 120. Therefore, no copies of the cited documents are included herewith pursuant to 37 C.F.R. § 1.98(d); MPEP § 609(III)(A)(2).

☒ are enclosed unless they are U.S. patents or U.S. patent application publications pursuant to 37 C.F.R. § 1.98(a).

I
BASIS

This information disclosure statement is filed pursuant to

☒ 37 C.F.R. § 1.97(b).
This information disclosure statement is filed either (1) within three months of the filing date of a national application other than a continued prosecution application under 37 C.F.R. § 1.53(d); (2) within three months of the date of entry of the national stage as set forth in 37 C.F.R. § 1.491 in an international application; (3) before the mailing date of a first Office action on the merits; (4) concurrently with the filing of a request for continued examination under 37 C.F.R. § 1.114; or (5) before the mailing of a first Office action after the filing of a request for continued examination under 37 C.F.R. § 1.114, whichever event occurs last.

Accordingly, this information disclosure statement requires no fee and no certification.

☐ 37 C.F.R. § 1.97(c).
This information disclosure statement is filed after the period specified in 37 C.F.R. § 1.97(b), but before the mailing date of any of (1) a final action under 37 C.F.R. § 1.113; (2) a notice of allowance under 37 C.F.R. § 1.311; or (3) an action that otherwise closes prosecution in the application.

Accordingly, this information disclosure statement requires either the fee specified in 37 C.F.R. § 1.17(p) for submission of an information disclosure statement under 37 C.F.R. § 1.97(c), or a statement under 37 C.F.R. § 1.97(e).

☐ 37 C.F.R. § 1.97(d).
This information disclosure statement is filed after the period specified in 37 C.F.R. § 1.97(c) since a:

☐ final action under 37 C.F.R. § 1.113

☐ notice of allowance under 37 C.F.R. § 1.311

was mailed to the Applicant on _____.

Accordingly, this information disclosure statement requires the fee set forth in 37 C.F.R. § 1.17(p) to consider an information disclosure statement under 37 C.F.R. § 1.97(d), and a statement under 37 C.F.R. § 1.97(e).

If this statement crosses in the mail with an Office action, or is otherwise not in the indicated category of 37 C.F.R. § 1.97, it is respectfully requested that this statement be treated in the next appropriate category and made of record. To the extent required, please treat this paper as a conditional petition for acceptance of the information disclosure statement.

II
FEES

Any required fee is being paid as indicated below:

☒ No fee is due.

☐ The $180 fee specified in 37 C.F.R. § 1.17(p) for submission of an information disclosure statement under 37 C.F.R. § 1.97(c) or (d) is due.

☐ This fee is paid by credit card.

☐ Please charge the required fee to Deposit Account No. 505063.

☒ The Commissioner is hereby authorized to charge any additional required fees or credit any overpayments associated with this information disclosure statement to the account.

III

CERTIFICATION

Pursuant to 37 C.F.R. § 1.97(e), I state as follows:

☒ No statement is necessary.

☐ Each item of information contained in this information disclosure statement was first cited in a communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of this information disclosure statement. 37 C.F.R. § 1.97(e)(1).

☐ No item of information contained in this information disclosure statement was cited in a communication from a foreign patent office in a counterpart foreign application, and, to my knowledge after making reasonable inquiry, no item of information contained in this information disclosure statement was known to any individual designated in 37 C.F.R. § 1.56(c) more than three months prior to the filing of this information disclosure statement. 37 C.F.R. § 1.97(e)(2).

Pursuant to 37 C.F.R. § 1.704(d), I state as follows:

☐ Each item of information contained in this information disclosure statement was cited in a communication from a foreign patent office in a counterpart application, and this communication was not received by any individual designated in 37 C.F.R. § 1.56(c) more than thirty days prior to the filing of this information disclosure statement.

IV
FOREIGN LANGUAGE REFERENCES

Pursuant to 37 C.F.R. § 1.98(a)(3)(i), any information disclosure statement filed under 37 C.F.R. § 1.97 shall include a concise explanation of the relevance, as it is presently understood by the individual designated in § 1.56(c) most knowledgeable about the content of the information, of each patent, publication, or other information listed that is not in the English language. The concise explanation may be either separate from the specification or incorporated therein. In compliance with the rules, the Applicant states as follows:

☐ No concise explanation is required since all cited references are in English.

☐ A concise explanation of the relevance of one or more foreign language references is attached.

☒ According to the procedural policy adopted by the PTO with regard to disclosure requirements, the Applicant is satisfying the disclosure requirement pursuant to MPEP § 609.04(a)(III) by submitting herewith the following:

- ☐ English language versions of any non-English language documents.
- ☐ English language versions of one or more search reports or other actions by one or more foreign patent offices in one or more counterpart foreign applications, which indicate the degree of relevance found by the foreign office(s).
- ☒ Pursuant to 37 C.F.R. § 1.98(a)(3)(ii), a copy of a translation is not submitted herewith, *for the below listed references*, because a written English-language translation of a non-English-language document, or portion thereof, is not within the possession, custody, or control of, or is readily available to any individual designated in § 1.56(c).

Publication No.	Pub. Date	Applicant
JP-05162529	Jun 29, 1993	Hiroshi, Saito
JP-07066829	Mar 10, 1995	Hiroshi, Saito
JP-08328583	Dec 13, 1996	NT&T
JP-10027148A2	Jan 27, 1998	ONO, S. et al.
CN-1264476A	Aug 23, 2000	Kenner, Brian et al.
WO-199859486	Dec 30, 1998	Telecom PTT
JP-2000207270	Jul 28, 2000	Tomoya, Anzai et al.
KR-20010023599	Mar 26, 2001	Malkin et al.
JP-2001007844	Jan 12, 2001	Tsukasa, Ogino
JP-2001290787	Oct 19, 2001	Hidenori, Araki et al.
JP-2001312434	Nov 9, 2001	Toru, Okugawa

The Applicant has not analyzed these documents and, therefore, declines to comment on their relevance at this time.

☐ One or more English language abstracts for the non-English language references are also enclosed.

U.S. Patent counterparts have been (or are being) cited in lieu of copies of the following cited foreign patents[1]:

Foreign Publication	Pub Date	Applicant	Also published as:
EP-1104555	Jun 6, 2001	IBM	US6263313 US6611812
IL-140935	Mar 19, 2006	IBM	US6587837 US6418421 US6398245 US6263313
JP-2002522995	Jul 23, 2002	IBM	US6263313 US6418421
JP-2005124165	May 12, 2005	IBM	US6263313 US6418421
JP-3566626	Jun 18, 2004	IBM	US6460082
JP-3762649	Jan 20, 2006	IBM	US6763377
AU-763380	Jun 11, 2003	IBM	US6226618
TW-NI176482	Aug 19, 2003	IBM	US6418421 US6226618
CN-ZL99810853.7	Aug 12, 1999	IBM	US6587837 US6418421 US6398245 US6263313 CN1320232

V
REMARKS

It is respectfully requested that:

(1) The Examiner sign the enclosed Form to evidence that the cited information has been fully considered by the Patent and Trademark Office during the examination of this application; and

(2) The citations for the cited information be printed on any patent that issues from this application.

The submission of this information disclosure statement by the Applicant shall not be construed as a representation that a search has been performed or that no other material information under 37 C.F.R. § 1.56 exists. See 37 C.F.R. § 1.97(g).

The submission of this information disclosure statement by the Applicant shall not be construed as an admission that any information cited or referred to herein is, or is considered to be, material to the patentability of one or more of the pending claims in this application pursuant

[1] According to European Patent Office (per website http://ep.espacenet.com)

to 37 C.F.R. § 1.56(b), unless specifically designated by the Applicant as such. See 37 C.F.R. § 1.97(h).

The submission of this information disclosure statement by the Applicant shall not be construed as an admission that any information cited or referred to herein is, or is considered to be, or even qualifies as "prior art" under 35 U.S.C. § 102 with respect to this invention, unless specifically designated by the Applicant as such.

Notwithstanding any statements by the Applicant, the Examiner is urged to form his or her own conclusion regarding the relevance of the cited information.

VI
CONCLUSION

The Examiner is encouraged to contact the undersigned attorney to discuss any questions concerning this information disclosure statement or any of the information cited or referred to herein.

Respectfully submitted,

Date: October 21, 2014

/JCS/

Jonathan C. Siekmann
Registration No. 58,259
USPTO Customer No. 83579
Level 3 Communications, LLC
1025 Eldorado Blvd.
Broomfield, Colorado 80021
Tel: 720-888-2140
Fax: 720-888-5619

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	1	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		20010029525	Oct 11, 2001	Lahr, Nils B.	
		20010049732A1	Dec 6, 2001	Raciborski, N. F., et al.	
		20010051980	Dec 13, 2001	Raciborski, Nathan F., et al.	
		20010056500	Dec 27, 2001	Farber, David A., et al.	
		20020018449	Feb 14, 2002	Ricciulli, Livio	
		20020023164	Feb 14, 2002	Lahr, Nils B.	
		20020023165	Feb 21, 2002	Lahr, Nils B.	
		20020042817	Apr 11, 2002	Lahr, Nils B.	
		20020046273	Apr 18, 2002	Lahr, Nils B., et al.	
		20020046405	Apr 18, 2002	Lahr, Nils B.	
		20020049857	Apr 25, 2002	Farber, David A., et al.	
		20020059592	May 16, 2002	Kiraly, Jozsef	
		20020066038	May 30, 2002	Mattsson, Ulf	
		20020073199	Jun 13, 2002	Levine, Matthew et al.	
		20020078233	Jun 20, 2002	Biliris, Alexandros et al.	
		20020082999	Jun 27, 2002	Lee, Cheol-Woong et al.	
		20020083124	Jun 27, 2002	Knox, Christopher R., et al.	
		20020089470	Jul 11, 2002	Raman, Mithila et al.	
		20020092026	Jul 11, 2002	Janniello, James P., et al.	
		20020099850	Jul 25, 2002	Farber, David A., et al.	
		20020124080	Sep 5, 2002	Leighton, F. T.	
		20020129134	Sep 12, 2002	Leighton, F. T., et al.	
		20020131645	Sep 19, 2002	Hamilton, Chris H.	
		20020143798	Oct 3, 2002	Lisiecki, Philip A., et al.	
		20020143888	Oct 3, 2002	Lisiecki, Philip A., et al.	
		20020147774	Oct 10, 2002	Lisiecki, Philip A., et al.	
		20020152318	Oct 17, 2002	Menon, Satish N., et al.	
		20020163882	Nov 7, 2002	Bornstein, Claudson F., et al.	
		20020199016	Dec 26, 2002	Freedman, Avraham T.	
		20030009444	Jan 9, 2003	Eidler, Christopher W., et al.	
		20030018966	Jan 23, 2003	Cook, David H., et al.	
		20030028623	Feb 6, 2003	Hennessey, Wade L., et al.	
		20030028626	Feb 6, 2003	Hennessey, Wade L., et al.	
		20030028777	Feb 6, 2003	Hennessey, Wade L., et al.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	2	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		20030055972	Mar 20, 2003	Fuller, William T., et al.	
		20030061263	Mar 27, 2003	Riddle, Guy	
		20030061280	Mar 27, 2003	Bulson, Jason A., et al.	
		20030065703	Apr 3, 2003	Aborn, Justin	
		20030065762	Apr 3, 2003	Stolorz, Paul E., et al.	
		20030078888	Apr 24, 2003	Lee, Cheol-Woong et al.	
		20030078889	Apr 24, 2003	Lee, Cheol-Woong et al.	
		20030079027	Apr 24, 2003	Slocombe, Michael et al.	
		20030095660	May 22, 2003	Lee, Chang Y., et al.	
		20030105604	Jun 5, 2003	Ash, Leslie E., et al.	
		20030154239	Aug 14, 2003	Davis, Andrew T., et al.	
		20030158923	Aug 21, 2003	Burkhart, Reed	
		20030158928	Aug 21, 2003	Knox, Christopher R., et al.	
		20040022194	Feb 5, 2004	Ricciulli, Livio	
		20040139097	Jul 15, 2004	Farber, David A., et al.	
		20040177148	Sep 9, 2004	Tsimelzon Jr., Mark	
		20050033858	Feb 10, 2005	Swildens, Eric S., et al.	
		20050038851	Feb 17, 2005	Kriegsman, Mark E.	
		20050100027	May 12, 2005	Leinwand, Allan S., et al.	
		20050114296	May 26, 2005	Farber, David A., et al.	
		20050262104	Nov 24, 2005	Robertson, Niel et al.	
		20060143293	Jun 29, 2006	Freedman, Avraham T.	
		4495570	Jan 22, 1985	Kitajima, Hiroyuki et al.	
		4591983	May 27, 1986	Bennett, James S., et al.	
		4594704	Jun 10, 1986	Ollivier, Y.	
		4726017	Feb 16, 1988	Krum, H. et al.	
		4803641	Feb 7, 1989	Hardy, S. et al.	
		4839798	Jun 13, 1989	Eguchi, H. et al.	
		4847784	Jul 11, 1989	Clancey, W. J.	
		4920432	Apr 24, 1990	Eggers, D. et al.	
		4922417	May 1, 1990	Churm, B. et al.	
		4943932	Jul 24, 1990	Lark, J. S., et al.	
		4949187	Aug 14, 1990	Cohen, J. M.	
		4949248	Aug 14, 1990	Caro, M. A.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	3	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5029232	Jul 2, 1991	Nall, D. C.	
		5130792	Jul 14, 1992	Tindell, E. G., et al.	
		5132992	Jul 21, 1992	Yurt, P. et al.	
		5136716	Aug 4, 1992	Harvey, G. et al.	
		5172413	Dec 15, 1992	Bradley, G. C., et al.	
		5191573	Mar 2, 1993	Hair, A. R.	
		5253275	Oct 12, 1993	Yurt, P. et al.	
		5253341	Oct 12, 1993	Rozmanith, A. et al.	
		5287499	Feb 15, 1994	Nemes, R. M.	
		5287537	Feb 15, 1994	Newmark, R. J., et al.	
		5291554	Mar 1, 1994	Morales, F.	
		5341477	Aug 23, 1994	Pitkin, R. P., et al.	
		5371532	Dec 6, 1994	Gelman, A. et al.	
		5410343	Apr 25, 1995	Coddington, C. D., et al.	
		5414455	May 9, 1995	Hooper, D. F., et al.	
		5442389	Aug 15, 1995	Blahut, D. E., et al.	
		5442390	Aug 15, 1995	Hooper, D. F., et al.	
		5442749	Aug 15, 1995	Northcutt, J. D., et al.	
		5471622	Nov 28, 1995	Eadline, D. J.	
		5475615	Dec 12, 1995	Lin, K. D.	
		5508732	Apr 16, 1996	Bottomley, J. F., et al.	
		5515511	May 7, 1996	Nguyen, T. et al.	
		5519435	May 21, 1996	Anderson, M. H.	
		5528281	Jun 18, 1996	Grady, J. et al.	
		5539621	Jul 23, 1996	Kikinis, D.	
		5542087	Jul 30, 1996	Neimat, M. et al.	
		5544313	Aug 6, 1996	Shachnai, H. et al.	
		5544327	Aug 6, 1996	Dan, A. et al.	
		5550577	Aug 27, 1996	Verbiest, W. J., et al.	
		5550863	Aug 27, 1996	Yurt, P. et al.	
		5550982	Aug 27, 1996	Long, J. E., et al.	
		5557317	Sep 17, 1996	Nishio, M. et al.	
		5568181	Oct 22, 1998	Greenwood, David G., et al.	
		5572643	Nov 5, 1996	Judson, D. H.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	4	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5590288	Dec 31, 1996	Castor, P. F., et al.	
		5592611	Jan 7, 1997	Midgely, C. W., et al.	
		5594910	Jan 14, 1997	Filepp, R. et al.	
		5603026	Feb 11, 1997	Demers, A. J., et al.	
		5614940	Mar 25, 1997	Cobbley, David A., et al.	
		5619648	Apr 8, 1997	Canale, L. M., et al.	
		5623656	Apr 22, 1997	Lyons, K. B.	
		5625781	Apr 29, 1997	Cline, T. L., et al.	
		5627829	May 6, 1997	Gleeson, B. J., et al.	
		5630067	May 13, 1997	Kindell, C. N., et al.	
		5633999	May 27, 1997	Clowes, R. F., et al.	
		5634006	May 27, 1997	Baugher, M J., et al.	
		5638443	Jun 10, 1997	Stefik, M. J., et al.	
		5644714	Jul 1, 1997	Kikinis, D.	
		5646676	Jul 8, 1997	Dewkett, T. J., et al.	
		5649186	Jul 15, 1997	Ferguson, G. J.	
		5659729	Aug 19, 1997	Nielsen, J.	
		5666362	Sep 9, 1997	Chen, C. T., et al.	
		5671279	Sep 23, 1997	Elgamal, T.	
		5675734	Oct 7, 1997	Hair, A. R.	
		5682512	Oct 28, 1997	Tetrick, R. S.	
		5699513	Dec 16, 1997	Feigen, R. et al.	
		5712979	Jan 27, 1998	Graber, T. E., et al.	
		5715453	Feb 3, 1998	Stewart, G.	
		5721914	Feb 24, 1998	DeVries, L. R.	
		5734719	Mar 31, 1998	Tsevdos, James T., et al.	
		5734831	Mar 31, 1998	Sanders, J. B.	
		5740423	Apr 14, 1998	Logan, J. R., et al.	
		5742762	Apr 21, 1998	Scholl, Thomas H., et al.	
		5751961	May 12, 1998	Smyk, D.	
		5761507	Jun 2, 1998	Govett, I. R.	
		5761663	Jun 2, 1998	Lagarde, K. C., et al.	
		5764906	Jun 9, 1998	Edelstein, M. et al.	
		5774660	Jun 30, 1998	Brendel, J. et al.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	5	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5774668	Jun 30, 1998	Choquier, P. et al.	
		5777989	Jul 7, 1998	McGarvey, J.	
		5778187	Jul 7, 1998	Monteiro, Antonio M., et al.	
		5781909	Jul 14, 2009	Logan, James et al.	
		5784058	Jul 21, 1998	LaStrange, T. E., et al.	
		5796952	Aug 18, 1998	Davis, O. et al.	
		5799141	Aug 25, 1998	Galipeau, K. J., et al.	
		5802106	Sep 1, 1998	Packer, R. L.	
		5802291	Sep 1, 1998	Balick, M. et al.	
		5812769	Sep 22, 1998	Graber, T. E., et al.	
		5815664	Sep 29, 1998	Asano, K.	
		5819092	Oct 6, 1998	Ferguson, Charles H., et al.	
		5828847	Oct 27, 1998	Gehr, C. R., et al.	
		5832069	Nov 3, 2009	Waters, David B., et al.	
		5832506	Nov 3, 1998	Kuzma, A. J.	
		5832514	Nov 3, 1998	Norin, S. et al.	
		5835718	Nov 10, 1998	Blewett, C. D.	
		5845090	Dec 1, 1998	Collins, Theodore J., et al.	
		5845303	Dec 1, 1998	Templeman, M. L.	
		5856974	Jan 5, 1999	Gervais, J. L., et al.	
		5862339	Jan 19, 1999	Bonnaure, P. P., et al.	
		5867706	Feb 2, 1999	Martin, Sean J., et al.	
		5867799	Feb 2, 1999	Lang, A. K., et al.	
		5870546	Feb 9, 1999	Kirsch, S. T.	
		5870559	Feb 9, 1999	Leshem, E. et al.	
		5878212	Mar 2, 1999	Civanlar, S. et al.	
		5884038	Mar 16, 1999	Kapoor, R.	
		5890171	Mar 30, 1999	Blumer, T. P., et al.	
		5893116	Apr 6, 1999	Simmonds, C. D., et al.	
		5894554	Apr 13, 1999	Lowery, K. et al.	
		5896533	Apr 20, 1999	Ramos, D. et al.	
		5898456	Apr 27, 1999	Wahl, S.	
		5903723	May 11, 1999	Beck, R. D., et al.	
		5907704	May 25, 1999	Gudmundson, N. K., et al.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	6	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		5913028	Jun 15, 1999	Wang, F. et al.	
		5913033	Jun 15, 1999	Grout, G. B.	
		5918010	Jun 29, 1999	Appleman, K. H., et al.	
		5919247	Jul 6, 1999	Van, H. A., et al.	
		5920701	Jul 6, 1999	Miller, C. K., et al.	
		5931904	Aug 3, 1999	Banga, Gaurav	
		5933832	Aug 3, 1999	Suzuoka, T. et al.	
		5933835	Aug 3, 1999	Adams, Robert et al.	
		5935207	Aug 10, 1999	Logue, J. D., et al.	
		5940831	Aug 17, 1999	Takano, H.	
		5944780	Aug 31, 1999	Chase, J. S.	
		5951694	Sep 14, 1999	Choquier, P.	
		5956489	Sep 21, 1999	San Andres, R. J., et al.	
		5956716	Sep 21, 1999	Kenner, B. et al.	
		5958008	Sep 28, 1999	Pogrebisky, M. et al.	
		5961596	Oct 5, 1999	Takubo, S. et al.	
		5966440	Oct 12, 1999	Hair, A. R.	
		5968121	Oct 19, 1999	Logan, K. W., et al.	
		5973696	Oct 26, 1999	Agranat, I. D., et al.	
		5978791	Nov 2, 1999	Farber, D. A., et al.	
		5983214	Nov 9, 1999	Lang, A. K., et al.	
		5983227	Nov 9, 1999	Nazem, F. et al.	
		5987606	Nov 16, 1999	Cirasole, P. et al.	
		5991809	Nov 23, 1999	Kriegsman, M. E., et al.	
		5996025	Nov 30, 1999	Day	
		6002720	Dec 14, 1999	Yurt, P. et al.	
		6003030	Dec 14, 1999	Kenner, B. et al.	
		6012090	Jan 4, 2000	Chung, Pi-Yu. et al.	
		6014686	Jan 11, 2000	Elnozahy, E. N., et al.	
		6014698	Jan 11, 2000	Griffiths, M. J.	
		6016509	Jan 18, 2000	Dedrick, Rick	
		6016512	Jan 18, 2000	Huitema, Christian	
		6018516	Jan 25, 2000	Packer, R. L.	
		6021426	Feb 1, 2000	Douglis, F.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	7	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6023470	Feb 8, 2000	Lee, Warren S., et al.	
		6029175	Feb 22, 2000	Chow, Yen-whei et al.	
		6029176	Feb 22, 2000	Cannon, M. E.	
		6035332	Mar 7, 2000	Ingrassia, M. I., et al.	
		6038216	Mar 14, 2000	Packer, R. L.	
		6038310	Mar 14, 2000	Hollywood, S. et al.	
		6041324	Mar 21, 2000	Earl, J. R., et al.	
		6044405	Mar 28, 2000	Driscoll, E. J., et al.	
		6046980	Apr 4, 2000	Packer, R. L.	
		6049831	Apr 11, 2000	Gardell, S. E., et al.	
		6052718	Apr 18, 2000	Gifford, D. K.	
		6052730	Apr 18, 2000	Felciano, R. M., et al.	
		6065051	May 16, 2000	Steele, D. W., et al.	
		6065062	May 16, 2000	Periasamy, R. et al.	
		6070191	May 30, 2000	Narendran, B. et al.	
		6078943	Jun 20, 2000	Yu, P. S.	
		6081829	Jun 27, 2000	Sidana, A. S.	
		6081835	Jun 27, 2000	Antcliff	
		6081840	Jun 27, 2000	Zhao, Yan	
		6092112	Jul 18, 2000	Fukushige, Y.	
		6092204	Jul 18, 2000	Baker, B. S., et al.	
		6094706	Jul 25, 2000	Factor, Michael E., et al.	
		6098078	Aug 1, 2000	Gehani, N. H.	
		6098096	Aug 1, 2000	Tsirigotis, Panagiotis et al.	
		6105028	Aug 15, 2000	Sullivan, D. J., et al.	
		6108673	Aug 22, 2000	Brandt, M. L., et al.	
		6108703	Aug 22, 2000	Leighton, F. T., et al.	
		6112231	Aug 29, 2000	DeSimone, A. et al.	
		6112239	Aug 29, 2000	Kenner, B. et al.	
		6112240	Aug 29, 2000	Pogue, M. A., et al.	
		6115357	Sep 5, 2000	Packer, R. L., et al.	
		6115752	Sep 5, 2000	Chauhan, A.	
		6119143	Sep 12, 2000	Dias, D. M., et al.	
		6125388	Sep 26, 2000	Reisman, R. R.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	8	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6125394	Sep 26, 2000	Rabinovich, Michael	
		6128601	Oct 3, 2000	Van Horne, P. et al.	
		6128623	Oct 3, 2000	Mattis, Peter et al.	
		6128660	Oct 3, 2000	Grimm, S. M., et al.	
		6130890	Oct 10, 2000	Leinwand, A. S., et al.	
		6131095	Oct 10, 2000	Low, Colin et al.	
		6134583	Oct 17, 2000	Herriot, R. G.	
		6144375	Nov 7, 2000	Jain, R. C., et al.	
		6144702	Nov 7, 2000	Yurt, P. et al.	
		6151624	Nov 21, 2000	Teare, K. et al.	
		6154744	Nov 28, 2000	Kenner, B. et al.	
		6154753	Nov 28, 2000	McFarland, J. A.	
		6161137	Dec 12, 2000	Johnson, Frank E., et al.	
		6163779	Dec 19, 2000	Mantha, M. L., et al.	
		6173311	Jan 9, 2001	Hassett, G. P., et al.	
		6173322	Jan 9, 2001	Hu, W. M.	
		6178160	Jan 23, 2001	Bolton, D. W., et al.	
		6181867	Jan 30, 2001	Kenner, B. et al.	
		6185598	Feb 6, 2001	Farber, David A., et al.	
		6185619	Feb 6, 2001	Joffe, R. L., et al.	
		6189030	Feb 13, 2001	Kirsch, S. T., et al.	
		6189039	Feb 13, 2001	Harvey	
		6195680	Feb 27, 2001	Goldszmidt, German S., et al.	
		6205120	Mar 20, 2001	Packer, R. L., et al.	
		6226618	May 1, 2001	Downs	
		6226642	May 1, 2001	Beranek, M. J., et al.	
		6230196	May 8, 2001	Guenthner, T. J., et al.	
		6233623	May 15, 2001	Jeffords, Jason et al.	
		6240462	May 29, 2001	Agraharam, Sanjay et al.	
		6249810	Jun 19, 2001	Kiraly, J.	
		6256675	Jul 3, 2001	Rabinovich, M.	
		6263313	Jul 17, 2001	Milsted	
		6266335	Jul 24, 2001	Bhaskaran, Sajit	
		6266699	Jul 24, 2001	Sevcik, M.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	9	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6269394	Jul 31, 2001	Kenner, B. et al.	
		6272566	Aug 7, 2001	Craft	
		6275470	Aug 14, 2001	Ricciulli, L.	
		6282569	Aug 28, 2001	Wallis, G. D., et al.	
		6286045	Sep 4, 2001	Griffith, M. J., et al.	
		6298041	Oct 2, 2001	Packer, R. L.	
		6314565	Nov 6, 2001	Kenner, B. et al.	
		6317787	Nov 13, 2001	Boyd, William G., et al.	
		6324580	Nov 27, 2001	Jindal, Anita et al.	
		6324582	Nov 27, 2001	Sridhar, Manickam R., et al.	
		6330602	Dec 11, 2001	Law, E. K., et al.	
		6332157	Dec 18, 2001	Mighdoll, Lee S., et al.	
		6338044	Jan 8, 2002	Cook, D. et al.	
		6343298	Jan 29, 2002	Savchenko, Alexander V., et al.	
		6345294	Feb 5, 2002	O'Toole, James et al.	
		6347085	Feb 12, 2002	Kelly, K. C., et al.	
		6351775	Feb 26, 2002	Yu, P. S.	
		6351776	Feb 26, 2002	O'Brien, Brett et al.	
		6360256	Mar 19, 2002	Lim, S. B.	
		6370571	Apr 9, 2002	Medin, M. S.	
		6370580	Apr 9, 2002	Kriegsman, M. E.	
		6389462	May 14, 2002	Cohen, Ariel et al.	
		6398245	Jun 4, 2002	Gruse	
		6405252	Jun 11, 2002	Gupta, A. K., et al.	
		6405257	Jun 11, 2002	Gersht, A. et al.	
		6412000	Jun 25, 2002	Riddle, G. et al.	
		6415280	Jul 2, 2002	Farber, D. A., et al.	
		6415368	Jul 2, 2002	Glance, Natalie S., et al.	
		6418421	Jul 9, 2002	Hurtado	
		6421714	Jul 16, 2002	Rai, Girish et al.	
		6421726	Jul 16, 2002	Kenner, B. et al.	
		6427170	Jul 30, 2002	Sitaraman, Aravind et al.	
		6430618	Aug 6, 2002	Karger, D. et al.	
		6442588	Aug 27, 2002	Clark, Neal H., et al.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	10	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6453319	Sep 17, 2002	Mattis, Peter et al.	
		6456630	Sep 24, 2002	Packer, R. L., et al.	
		6460082	Oct 1, 2002	Lumelsky	
		6460085	Oct 1, 2002	Toporek, J. D., et al.	
		6463508	Oct 8, 2002	Wolf	
		6466949	Oct 15, 2002	Yang, Mike et al.	
		6470389	Oct 22, 2002	Chung, Pi-Yu et al.	
		6473405	Oct 29, 2002	Ricciulli, L.	
		6480893	Nov 12, 2002	Kriegsman, M. E.	
		6484143	Nov 19, 2002	Swildens, E. S., et al.	
		6484204	Nov 19, 2002	Rabinovich, M.	
		6487555	Nov 26, 2002	Bharat, Krishna A., et al.	
		6490580	Dec 3, 2002	Dey, J. K., et al.	
		6493707	Dec 10, 2002	Dey, J. K., et al.	
		6496856	Dec 17, 2002	Kenner, B. et al.	
		6502125	Dec 31, 2002	Kenner, B. et al.	
		6502205	Dec 31, 2002	Yanai, Moshe et al.	
		6502215	Dec 31, 2002	Raad, G. B., et al.	
		6505248	Jan 7, 2003	Casper, B. J., et al.	
		6529477	Mar 4, 2003	Toporek, J. D., et al.	
		6553413	Apr 22, 2003	Leighton, F T., et al.	
		6553420	Apr 22, 2003	Karger, D. et al.	
		6557054	Apr 29, 2003	Reisman, R. R.	
		6564251	May 13, 2003	Katariya, S. et al.	
		6577595	Jun 10, 2003	Counterman, R. C.	
		6581090	Jun 17, 2003	Lindbo, S. et al.	
		6584083	Jun 24, 2003	Toporek, J. D., et al.	
		6587837	Jul 1, 2003	Spagna	
		6591299	Jul 8, 2003	Riddle, G. et al.	
		6611862	Aug 26, 2003	Reisman, R. R.	
		6625643	Sep 23, 2003	Colby, K. W., et al.	
		6651141	Nov 18, 2003	Adrangi, Farid	
		6654344	Nov 25, 2003	Toporek, J. D., et al.	
		6654807	Nov 25, 2003	Farber, David A., et al.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	11	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6658464	Dec 2, 2003	Reisman, R. R.	
		6665706	Dec 16, 2003	Kenner, B. et al.	
		6678659	Jan 13, 2004	Van Kommer, Robert	
		6691148	Feb 10, 2004	Zinky, J. A., et al.	
		6694358	Feb 17, 2004	Swildens, E. S., et al.	
		6699418	Mar 2, 2004	Okada, K. et al.	
		6708137	Mar 16, 2004	Carley, S.	
		6718328	Apr 6, 2004	Norris, C. L.	
		6741563	May 25, 2004	Packer, R. L.	
		6751673	Jun 15, 2004	Shaw, D. M.	
		6754699	Jun 22, 2004	Swildens, E. S., et al.	
		6754706	Jun 22, 2004	Swildens, E. S., et al.	
		6763377	Jul 13, 2004	Belknap	
		6763388	Jul 13, 2004	Tsimelzon, M.	
		6778502	Aug 17, 2004	Ricciulli, L.	
		6799221	Sep 28, 2004	Kenner, B. et al.	
		6801576	Oct 5, 2004	Haldeman, R. M., et al.	
		6834306	Dec 21, 2004	Tsimelzon, M.	
		6842604	Jan 11, 2005	Cook, D. et al.	
		6859791	Feb 22, 2005	Spagna	
		6870851	Mar 22, 2005	Leinwand, A. S., et al.	
		6874032	Mar 29, 2005	Gersht, A. et al.	
		6901604	May 31, 2005	Kiraly, J.	
		6915307	Jul 5, 2005	Mattis, Peter et al.	
		6915329	Jul 5, 2005	Kriegsman, M. E.	
		6928442	Aug 9, 2005	Farber, D. A., et al.	
		6934255	Aug 23, 2005	Toporek, J. D., et al.	
		6950623	Sep 27, 2005	Brown, J. W., et al.	
		6963910	Nov 8, 2005	Belknap, William R., et al.	
		6963980	Nov 8, 2005	Mattsson, U.	
		6963981	Nov 8, 2005	Bailey, R A., et al.	
		6970432	Nov 29, 2005	Hankins, S. A., et al.	
		6973490	Dec 6, 2005	Robertson, N. et al.	
		6981050	Dec 27, 2005	Tobias, M. et al.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	12	of	21	Attorney Docket No: 0137-US-U1

US PATENT DOCUMENTS

Examiner Initial *	Cite No	Document Number	Publication Date	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear
		6981180	Dec 27, 2005	Bailey, S. T., et al.	
		6996616	Feb 7, 2006	Leighton, F, T., et al.	
		7003572	Feb 21, 2006	Lownsbrough, D. L., et al.	
		7007089	Feb 28, 2006	Freedman, A. T.	
		7010578	Mar 7, 2006	Lewin, D. M., et al.	
		7012900	Mar 14, 2006	Riddle, G.	
		7013342	Mar 14, 2006	Riddle, Guy	
		7032072	Apr 18, 2006	Quinn, Michael J., et al.	
		7039633	May 2, 2006	Dey, J. K., et al.	
		7047300	May 16, 2006	Oehrke, T. L., et al.	
		7054902	May 30, 2006	Toporek, Jerome D., et al.	
		7054935	May 30, 2006	Farber, David A., et al.	
		7058706	Jun 6, 2006	Iyer, Shankar et al.	
		7069177	Jun 27, 2006	Carley, Stephen	
		7096266	Aug 22, 2006	Lewin, Daniel M., et al.	
		7103564	Sep 5, 2006	Ehnebuske	
		7110984	Sep 19, 2006	Spagna	
		7117259	Oct 3, 2006	Rohwer	
		7188085	Mar 6, 2007	Pelletier	
		7206748	Apr 17, 2007	Gruse	

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No	Foreign Patent Document	Publication Date	Name of Patentee or Applicant of cited Document	Pages, Columns, Lines, Where Relevant Passages or RelevantFigures Appear	T[2]
		JP-05162529	Jun 29, 1993	Hiroshi, Saito		
		EP-0649121	Apr 19, 1995	IBM		
		EP-0651554	May 3, 1995	Eastman Kodak Co.		
		JP-07066829	Mar 10, 1995	Hiroshi, Saito		
		EP-0801487A2	Oct 15, 1997	Douglas, B. C.		

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	13	of	21	Attorney Docket No: 0137-US-U1

		EP-0824236A2	Feb 18, 1998	Baird, R. et al.		
		JP-08328583	Dec 13, 1996	NT&T		
		EP-0865180A2	Sep 16, 1998	Chung, P. Y., et al.		
		JP-10027148A2	Jan 27, 1998	ONO, S. et al.		
		JP-10093552	Apr 10, 1998	MURAKAMI, H. et al.		
		EP-1063831A2	Dec 27, 2000	Ogino, T.		
		EP-1104555	Jun 6, 2001	IBM		
		CN-1264476A	Aug 23, 2000	Kenner, Brian et al.		
		IL-140935	Mar 19, 2006	IBM		
		WO-1996/042041A2	Dec 27, 1996	LEVERGOOD, T. M., et al.		
		WO-1997/011429A1	Mar 27, 1997	GRABER, T. E., et al.		
		WO-1997/029423A1	Aug 14, 1997	STANFORD, C. et al.		
		WO-199742582	Nov 13, 1997	Netcast Communications Corp		
		WO-1998/004985A1	Feb 5, 1998	KRIEGSMAN, M.		
		WO-1998/006033A1	Feb 12, 1998	AGRANAT, I. D., et al.		
		WO-199859486	Dec 30, 1998	Telecom PTT		
		WO-199917227	Apr 8, 1999	IBM		
		WO-199948246	Sep 23, 1999	Reuters America Inc		
		WO-199953422	Oct 21, 1999	Inktomi Corp		
		WO-2000/029990A1	May 25, 2000	LEWIS, K. T., et al.		
		WO-2000060861	Oct 12, 2000	Armstrong, J. B.		
		JP-2000207270	Jul 28, 2000	Tomoya, Anzai et al.		
		KR-20010023599	Mar 26, 2001	Malkin et al.		
		JP-2001007844	Jan 12, 2001	Tsukasa, Ogino		
		JP-2001290787	Oct 19, 2001	Hidenori, Araki et al.		
		JP-2001312434	Nov 9, 2001	Toru, Okugawa		
		WO-200193533	Dec 6, 2001	Aerocast Com Inc		
		JP-2002522995	Jul 23, 2002	IBM		
		JP-2005124165	May 12, 2005	IBM		
		CA-2202572	Oct 14, 1998	NANDY, B. et al.		
		GB-2281793A1	Mar 15, 1995	GRAHAM, D. W., et al.		

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. 1 Applicant's unique citation designation number (optional) 2 Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	14	of	21	Attorney Docket No: 0137-US-U1

		CA-2288488	Jun 16, 2000	Cohen, Ariel et al.		
		CA-2335661	Mar 9, 2001	IBM		
		CA-2335662	Mar 9, 2001	IBM		
		GB-2353877	Apr 28, 2004	IBM		
		CA-2467998	Apr 25, 2006	IBM		
		JP-3566626	Jun 18, 2004	IBM		
		JP-3762649	Jan 20, 2006	IBM		
		AU-763380	Jun 11, 2003	IBM		
		TW-NI176482	Aug 19, 2003	IBM		
		CN-ZL99810853.7	Aug 12, 1999	IBM		

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		"Exporting Web Server Final Report", http://www.cs.technion.ac.il/Labs/Lccn/projects/spring97/project4/final_report.html (downloaded 07/07/07). Spring 1997	
		"IBM Technical Disclosure Bulletin; Local Area Network Server Replacement Procedure", Vol. 38, No.1 (January 1995) , 235-236	
		JP 10(1998) - 27148 machine translation prepared by Japanese Patent Office (JPO), published Jan-27-98, Title: "Server System for Internet," Applicant: Hitachi , 23 pgs.	
		Patent Abstracts of Japan "Server System for Internet", Pub. No. 10-027148, pub. date Jan-27-98, Applicant: Hitachi, computer translation , 12 pgs.	
		Patent Abstracts of Japan, "Electronic Mail Multiplexing System and Communication Control Method in the System", (Appln. No. JP19930162529), (06/30/1993) (Pub. No. JP 7066829)	
		Patent Abstracts of Japan, "Method and Device for Repeating and Converting Information", (Appln. No. JP19960328583) (Pub. No. JP10171727) 06-26-1998	
		ADLER, R. M., "Distributed Coordination Models for Client/Server Computing", Computer 28, 4 April 1995 , 14-22	
		AGGARWAL, A. et al., "Performance of Dynamic Replication Schemes for an Internet Hosting Service", Technical Report, AT&T Labs October 1998	
		ALMEROTH, K. et al., "Scalable Delivery of Web Pages Using Cyclic Best-Effort (UDP) Multicast", IEEE INFOCOM San Francisco, California, USA June 1998 , pp. 1-24	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	15	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		ANDRESEN, D. et al., "Multiprocessor scheduling with client resources to improve the response time of WWW applications", ACM Press, NY, Proc. 11th Inti Conf. on Supercomputing (Austria, ICS '97) July 1997 , 92-99	
		ANDRESEN, et al., "SWEB: Towards a Scalable World Wide Web Server on Multicomputers", Proc. IPPS 4/15/1996 , 850-856	
		AWERBUCH, et al., "Competitive Distributed File Allocation", in Proc. of the 25th Ann. ACM Symp. on Theory of Computing May 1993 , pp. 164-173	
		AWERBUCH, B. et al., "Distributed Paging for General Networks", In Proc. of the 7th ACM-SIAM Symposium on Discrete Algorithms, January 1996 , pp. 574-583	
		BAENTSCH, M et al., "Enhancing the Web's Infrastructure: From Caching to Replication", IEEE Internet Computing, Mar. 1997 , 1(2):18-27	
		BAENTSCH, M. et al., "Introducing Application-Level Replication and Naming into Today's Web", Computer Networks and ISDN Systems Vol. 28, Nos. 7-11 May 1996 , pp. 921-930	
		BARTAL, Y. et al., "Competitive Algorithms for Distributed Data Management", 24th Annual ACM STOC, 5/92, Victoria, B.C. Canada	
		BASTURK, E. et al., "Using network layer anycast for load distribution in the Internet", Tech. Rep., IBM TJ. Watson Research Center July 1997 , 21 pgs	
		BERNERS-LEE, T. et al., RFC 1738--Uniform Resource Locators Dec 1994 , pp. 1-25	
		BERNERS-LEE, T. et al., "Hypertext Markup Language - 2.0", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1866 Nov 1995 , pp. 1-77	
		BERNERS-LEE, T. et al., "Hypertext Transfer Protocol -- HTTP/1.0", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1945 May 1996 , pp. 1-60	
		BERNERS-LEE, T. et al., "Hypertext Transfer Protocol -- HTTP/1.0", Internet Engineering Task Force (IETF), Internet Draft, draft-ietf-http-v10-spec-00.ps March 1995 , pp. 1-59	
		BERRA, P. B. et al., "Architecture for Distributed Database Systems", Computer Communications vol. 13, No. 4 May 1, 1990 , pp. 217-231	
		BESTAVROS, A., "Demand-Based Document Dissemination to Reduce Traffic and Balance Load in Distributed Information Systems", In Proc. IEEE Symp. on Parallel and Distributed Processing, San Antonio, TX Oct. 1995 , 8 pgs.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	16	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		BESTAVROS, et al., "Server-Initiated Document Dissemination for the WWW", IEEE Data Engineering Bulletin Sept. 1996 , 19(3): 3-11	
		BESTAVROS, A., "Speculative Data Dissermination and Service to Reduce Server Load Network Traffic and Service Time in Distributed Information Systems", In Proc. ICDE '96: The 1996 Int'l Conf. on Data Enqineering (Mar. 1996) , 4 Pages	
		BHATTACHARJEE, S. et al., "Application-Layer Anycasting", in Proc. IEEE INFOCOM '97 April 1997 , 9 pgs.	
		BRAUN, H. et al., "Web traffic characterization: an assessment of the impact of caching documents from NCSA's web server", Comput. Netw. ISDN Syst. 28, 1-2 Dec. 1995 , 37-51	
		BRISCO, T., "DNS Support for Load Balancing", IETF RFC 1794 April 1995 , pp. 1-7	
		CARTER, et al., "Dynamic server selection using bandwidth probing in wide-area networks", Tech. Rep. BU-CS-96-007, Compo Sci. Dept., Boston University Mar. 1996 , 1-20	
		CARTER, et al., "Server selection using dynamic path characterization in Wide-Area Networks", IEEE INFOCOM '97 1997 , pp. 1014-1021	
		CARTER, J. L. et al., "Universal Classes of Hash Functions", Journal of Computer and System Sciences Vol. 18, No.2 April 1979 , 106-112	
		CATE, V., "Alex: a global file system", in Proc. Usenix Conf. on File Systems May 1992 , pp. 1-11	
		CHANKHUNTHOD, A. et al., "A Hierarchical Internet Object Cache", Proc. Of the 1996 USENIX Technical Conf. Jan. 1996 , 153-163	
		COHEN, J. et al., "Cache Array Routing Protocol v1.1", http://tools.ietf.org/id/draft-vinod-carp-v1-01.txt (Last-Modified: Wed, 01 Oct 1997) 29 Sept. 1997 , 8 pages	
		COLAJANNI, et al., "Adaptive TTL schemes for load balancing of distributed Web servers", SIGMETRICS Perform. Eval. Rev. 25,2 Sep. 1997 , 36-42	
		COLAJANNI, M. et al., "Scheduling Algorithms for Distributed Web Servers", Intl Conf. on Distributed Computing Systems (ICDCS), Baltimore, Maryland, USA May 1997 , pp. 169-176	
		CROVELLA, et al., "Dynamic server selection in the Internet", 3rd IEEE Workshop on the Arch. and Implementation of High Performance Computer Sys. '95 Aug. 1995 , pp. 158-162	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	17	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		DANZIG, P. B. et al., "An analysis of wide-area name server traffic: a study of the Internet Domain Name System", Conf. Proc. Communications Architectures & Protocols; D. Oran, Ed. SIGCOMM '92; ACM Press, New York, NY Aug. 1992 , 281-292	
		DE BRA, P.M.E. et al., "Information Retrieval in the World Wide Web: Making Client-Based Searching Feasible", Computer Networks and ISDN System, NL, North Holland Publishing, Amsterdam, Vol. 27, No.2, ISSN: 0169-7552 11/01/1994 , 183-192	
		DEERING, S. E. et al., "Multicast routing in datagram internetworks and extended LANs", ACM Trans.Comput. Syst. 8,2 May. 1990 , 85-110	
		DOI, K., "Super Proxy Script -- How to make distributed proxy servers by URL hashing", Sharp Corp., http://naragw.sharp.co.jp/sps/; download 07/07/07. dates unknown (1996-2000)	
		ERIKSSON, H., "MBONE: The Multicast Backbone", Communications of the ACM vol. 37, no. 8 Aug 1994 , pp. 54-60	
		FEELEY, M. et al., "Implementing Global Memory Management in a Workstation Cluster", In Proc. 15th ACM Symp. on Operating Systems Principles Dec.1995 , 201-212	
		FIELDING, R. et al., "Hypertext Transfer Protocol -- HTTP/1.1", Internet Engineering Task Force (IETF) Internet Draft, draft-ietf-http-v11-spec-00.txt Nov 1995 , pp. 1-81	
		FIELDING, R. et al., "Hypertext Transfer Protocol -- HTTP/1.1", Internet Engineering Task Force (IETF) Request for Comments (RFC) 2068 Jan 1997 , pp. 1-162	
		FLOYD, S. et al., "A Reliable Multicast Framework for Light-Weight Sessions and Application Level Framing", In Proc. of ACM SIGCOMM '95, August, 1995 , 342-356	
		FOX, A. et al., "A Framework for Separating Server Scalability and Availability from Internet Application Functionality", PhD thesis, University of California, Berkeley, 1998 , 163 pgs	
		FOX, A. et al., "Cluster-based scalable network services", Proc. 16th ACM Symp. on Operating Systems Principles (Saint Malo, France, Oct. 05 - 08, 1997), W. M. Waite, Ed. SOSP '97. ACM Press, New York, NY , 78-91	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	18	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		GADDE, S. et al., "Reduce, reuse, recycle: An approach to building large internet caches", in Workshop on Hot Topics in Operating Systems (HotOS) April 1997 , pp. 93-98	
		GOLDSZMIDT, M. et al., "Load Distribution for Scalable Web Servers: Summer Olympics 1996--A Case Study", In Proc. 8th IFIPIIEEE Int'l Workshop on Distributed Systems: Operations and Management , Sydney, Australia. Oct. 1997 , 10 pgs.	
		GRIGNI, M. et al., "Tight Bounds on Minimum Broadcasts Networks", SIAM J. Disc. Math. 4 (May, 1991) , 207-222	
		GUYTON, et al., "Locating nearby copies of replicated Internet servers", Proc. ACM SIGCOMM '95; pp. 288-298 October 1995	
		GWERTZMAN, J. et al., "The Case for Geographical Push-Caching", Proc. Workshop on Hot OS '95 (05/04/1995) , 51-55	
		GWERTZMAN, J. et al., "World-Wide Web Cache Consistency", Proc. 1996 USENIX Tech. Conf., pgs.141-151, San Diego, CA Jan. 1996,	
		JEFFERY, C. et al., "Proxy sharing proxy servers", In Proc. IEEE etaCOM Conf. May 1996 , pp. 116-119	
		KARGER, D. et al., "Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web", In Proc. 29th Annual ACM Symp. on Theory of Computing May 1997 , 654-663	
		KIM, Y. J. et al., "Clustered multi-media NOD: Popularity-based article prefetching and placement", Sixteenth IEEE Symp. on Mass Storage Systems, San Diego, CA (March 15-18,1999 , pp. 194-202	
		KORKEA-AHO, M., "Scalability in Distributed Multimedia Systems", Technical report TKO-B128, Helsinki University of Technology	
		KRISHNAMURTHY, B. et al., "Study of piggyback cache validation for proxy caches in the World Wide Web", in: Symp. on Internet Technology and Systems, USENIX Association Dec. 1997	
		KROEGER, T. M. et al., "Exploring the Bounds of Web Latency Reduction from Caching and Prefetching", Proc. Usenix Symp. Internet Technologies and Systems, Usenix December 1997 , pp. 13-22	
		KWAN, et al., "NCSA's World Wide Web Server: Design and Performance", IEEE Nov. 1995 , pp. 68-74	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

Complete if Known	
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	19	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		LITTLE, T.D. C. et al., "Selection and Dissemination of Digital Video via the Virtual Video Browser", Multimedia Tools and Applications vol. 1, No. 2 (Netherlands) Jun. 1995 , pp. 149-172	
		LUOTONEN, et al., "World-Wide Web Proxies", CERN modified May 24,1994 Apr. 1994 , 1-8	
		MALKIN, G., "RIP Version 2 Carrying Additional Information", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1388 Jan 1993 , pp. 1-7	
		MALPANI, R. et al., "Making World Wide Web Caching Servers Cooperate", In Proc. 4th Int'l. World Wide Web Conf. (downloaded from http://www.w3.org/ConferencesIWWW4/Papers/59/ on 07/07/07) Dec. 1995 , 10 pages	
		MOCKAPETRIS, P., "Domain Names - Concepts and Facilities", IETF RFC 1034 Nov 1987 , 55 pgs.	
		MOCKAPETRIS, P., "Domain Names - Implementation and Specification", IETF RFC 1035 Nov 1987 , 55 pgs.	
		MOURAD, et al., "Scalable Web Server Architectures", iscc, 2nd IEEE Symposium on Computers and Communications (ISCC '97) July 1997 , pp. 12-16	
		MOY, J., "OSPF Version 2", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1583 March 1994 , 132 pgs.	
		NIKI, HITOSHI , "Storage Network", Computopia Vol. 36, No. 415, Japan, Computer Age Co., Ltd. (translation) April 1, 2001 , P. 52	
		OGUCHI, et al., "A Study of Caching Proxy Mechanisms Realized on Wide Area Distributed Networks", High Performance Distributed Computing, 5th Int'l Symposium August 1996 , pp. 443-449	
		PALMER, M. et al., "Fido: A Cache that Learns to Fetch", In Proc. the 17th Int'l Conf. on Very Large Data Bases Sept. 1991 , 255-264	
		PANIGRAHY, R., "Relieving Hot Spots on the World Wide Web", Master's thesis, MIT EECS June 1997 , pp. 1-66	
		PARRIS, C. et al., "A Dynamic Connection Management Scheme for Guaranteed Performance Services in Packet-Switching Integrated Services Networks", UC Berkeley Computer Science Division Tech. Report TR-93-005 1993 , 37 pgs.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)

* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	20	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		PARRIS, C. et al., "The Dynamic Management of Guaranteed Performance Connections in Packet Switched Integrated Service Networks", UC Berkeley Computer Science Division and International Computer Science Institute Tech. Report CSD-94-859 1994	
		PARTRIDGE, C. et al., "Host Anycasting Service", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1546 Nov 1993 , pp. 1-9	
		PETRI, S. et al., "Load Balancing and Fault Tolerance in Workstation Clusters. Migrating Groups of Communicating Processes", Operating Systems Review, Vol. 29, No.4 Oct. 1995 , 25-36	
		PLAXTON, G. C. et al., "Fast Fault-Tolerant Concurrent Access to Shared Objects", In Proc. 37th IEEE Symp. of Foundations of Computer Science Oct 1996 , pp. 570-579	
		POVEY, D. et al., "A distributed internet cache", in Proc. of the 20th Australasian Computer Science Conf. Feb. 1997	
		RABIN, M. O., "Efficient dispersal of information for security, load balancing, and fault tolerance", J.ACM 36, 2 (Apr. 1989) , pp. 335-348	
		RABINOVICH, M. et al., "Dynamic Replication on the Internet Work Project No. 3116-17-7006", AT&T Labs Research Mar. 5, 1998	
		RABINOVICH, M. et al., "RaDaR: A Scalable Architecture for a Global Web Hosting Service", WWW8 May 1999	
		REISSLEIN, M. et al., "Interactive video streaming with proxy servers", in Proc. of First International Workshop on Intelligent Multimedia Computing and Networking (IMMCN) 2000 , pp. 1-16	
		RODRIGUEZ, P. et al., "Improving the WWW: Caching or Multicast?", Computer Networks and ISDN Systems North Holland Publishing, Amsterdam, NL, vol. 30, No. 22-23, ISSN: 0169-7552 Nov. 1998 , pp. 2223-2243	
		ROSS, K. W., "Hash-Routing for Collections of Shared Web Caches", IEEE Network Magazine 11, 7:37-44 Nov-Dec 1997 , pp. 1-21	
		SEN, S. et al., "Proxy Prefix Caching for Multimedia Streams", IEEE June 1999 , 10 pgs.	
		SMITH, NEIL , "What can Archives offer the World Wide Web?", Technical Report 11, University of Kent, Computing Laboratory, University of Kent, Canterbury, UK March 1994 , 1-12	
		TAKAHASHI, TAKAO , "How to customize Apache Q&A -- Web server using Apache", AI Publishing, April 30, 1999, First Edition , pp. 63-64 (translation)	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

Substitute for form 1449A/PTO

INFORMATION DISCLOSURE STATEMENT BY APPLICANT

(Use as many sheets as necessary)

	Complete if Known
Application Number	12/408,681
Filing Date	Mar 21, 2009
First Named Inventor	David Fullagar
Art Unit	2452
Examiner Name	Thomas J. Dailey

Sheet	21	of	21	Attorney Docket No: 0137-US-U1

OTHER DOCUMENTS -- NON PATENT LITERATURE DOCUMENTS			
Examiner Initials*	**Cite No [1]**	**Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.**	**T[2]**
		THALER, D. G. et al., "Using name-based mappings to increase hit rates", IEEE/ACM Trans. Netw. 6,1 Feb. 1998 , 1-14	
		TOPOLCIC, C., "Experimental Internet Stream Protocol, Version 2 (ST-II)", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1190 Oct 1990 , pp. 1-148	
		TOUCH, J. et al., "The X-Bone", Third Global Internet Mini-Conference at Globecom '98, Sydney, Australia (pp. 44-52 of mini-conference) Nov 1998 , pp. 59-68	
		TRAINA, P., "BGP-4 Protocol Analysis", Internet Engineering Task Force (IETF) Request for Comments (RFC) 1774 March 1995 , pp. 1-10	
		VIN, H., "Multimedia Broadcasting Over the Internet: Part 1", IEEE Multimedia IEEE Computer Society, US, vol. 5, NR. 4 Oct. 1998 , pp. 78-82	
		VITTER, J. S. et al., "Optimal Prefetching via Data Compression", Proc. 32nd Annual IEEE Symposium on Foundations of Computer Science October 1991 , 21 pages	
		WESSELS, D., "Configuring Hierarchical Squid Caches", Aug. 19 1997 , 62 pgs.	
		WESSELS, DUANE , "Intelligent Caching for World-Wide Web Objects", Masters Thesis, University of Colorado (also presented at INET '95 in June '95) Jan 1995 , 1-85	
		WESSELS, D. et al., "Internet Cache Protocol (ICP), Version 2", Internet Engineering Task Force (IETF) Request for Comments (RFC) 2186 Sept 1997 , pp. 1-9	
		WOLFSON, O. et al., "An Adaptive Data Replication Algorith", ACM Transactions on Database Systems (TODS) Vol. 22(4) June 1997 , pp. 255-314	
		XIAO, LI et al., "Exploiting neglected data locality in browsers", in Proc. 10th Int'l World Wide Web Conference, (WWW10), Hong Kong (extended abstract) May 1-5 2001 , 2 pgs.	

EXAMINER **DATE CONSIDERED**

Substitute Disclosure Statement Form (PTO-1449)
* EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. [1] Applicant's unique citation designation number (optional) [2] Applicant is to place a check mark here if English language Translation is attached

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平5－162529

(43)公開日　平成5年(1993)6月29日

(51)Int.Cl.[5]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｂ６０Ｈ　1/00	１０１　Ｇ	7914－3L		
	１０３　Ｐ	7914－3L		

審査請求　未請求　請求項の数1(全 8 頁)

(21)出願番号　特願平3－331893

(22)出願日　平成3年(1991)12月16日

(71)出願人　000003137
マツダ株式会社
広島県安芸郡府中町新地3番1号

(72)発明者　土井　重紀
広島県安芸郡府中町新地3番1号　マツダ株式会社内

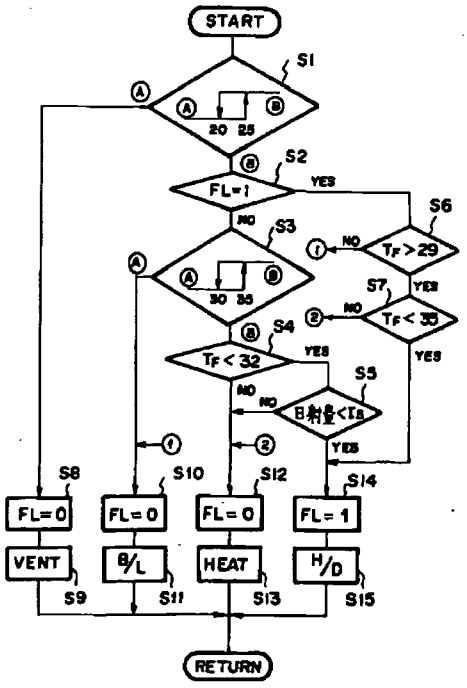
(74)代理人　弁理士　柳田　征史　(外1名)

(54)【発明の名称】　車両用空調装置

(57)【要約】

【目的】　吹出モードを必要吹出温度に基づいて自動制御する車両用空調装置において、中間季の夜間等において空調が涼しく感じるという問題を回避し、良好な空調を実現する。

【構成】　自動制御により選択可能な吹出モードとして少なくともヒートモードとバイレベルモードとヒートデフモードとを設定し、かつ同一の必要吹出温度においてはヒート吹出口からの吹出温度はヒートモードよりもヒートデフモードの方が高く、それよりも更にバイレベルモードの方が高く設定すると共に、ヒートモードにおけるヒート吹出口からの吹出温度よりもヒートデフモードにおけるデフロスト吹出口からの吹出温度のほうが低く、それよりも更にバイレベルモードにおけるベント吹出口からの吹出温度の方が低くなるように設定し、上記必要吹出温度がヒートモード要求温度範囲の内の低温側の所定温度範囲内であって日射量が所定値より少ない極少の場合は、ヒートモードからヒートデフモードに移行させる。



1

【特許請求の範囲】

【請求項1】　車室内外の熱環境に基づいて必要吹出温度を求め、該必要吹出温度に基づいて吹出モード及び吹出温度を自動制御する制御手段を備えて成る車両用空調装置であって、

上記吹出モードとして少なくともヒートモードとバイレベルモードとヒートデフモードとが設定されると共に、同一の必要吹出温度においてはヒート吹出口からの吹出温度は上記ヒートモードよりも上記ヒートデフモードの方が高く、それよりも更にバイレベルモードの方が高く設定され、ヒートモードにおけるヒート吹出口からの吹出温度よりも上記ヒートデフモードにおけるデフロスト吹出口からの吹出温度のほうが低く、それよりも更に上記バイレベルモードにおけるベント吹出口からの吹出温度の方が低くなるように設定され、

上記制御手段は、上記必要吹出温度がヒートモード要求温度範囲の内の低温側の所定温度範囲内でありかつ日射量が所定値より少ない極少の場合は、上記ヒートデフモードに移行させるものであることを特徴とする車両用空調装置。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、車室内外の熱環境を検出して必要吹出温度を求め、該必要吹出温度に基づいて吹出モード及び吹出温度を自動制御する制御手段を備えて成る車両用空調装置に関する。

【0002】

【従来の技術】従来より、例えば特開平2-26917 号公報に記載されているように、車室内外の熱環境例えば外気温、内気温、日射量、目標室内温度等に基づいて吹出モードや吹出温度等を自動制御する車両用空調装置が知られている。

【0003】かかる自動制御の一態様として、上記車室内外の熱環境に基づいて必要吹出温度（車室内を目標室内温度に維持するために必要な車室内への吹出空気の温度）を求め、該必要吹出温度に基づいて吹出温度（車室内に吹出す空気の温度）がその必要吹出温度になるようにエアミックスダンパの開度（この開度によりヒータコアを通る空気と通らない空気との割合が決まり、車室内に吹出す空気の温度が決まる）を制御すると共に、上記必要吹出温度に基づいて吹出モードを制御する、即ち必要吹出温度に応じて予め定められた吹出モードを選択するものが知られている。

【0004】

【発明が解決しようとする課題】ところで、上記のような従来の空調自動制御においては、例えば目標室内温度が通常の25℃の場合、中間季つまり春や秋（外気温が例えば15℃）の夜間等においては吹出モードとしてヒートモードが選択されるが、足元等に向けて吹出すヒート吹出口からの吹出温度がそれほど高くなく、その様な温度

2

が余り高くない空気の吹出しによって涼しく感じる状況が生じる。

【0005】この様な場合の対応策として、例えばエアミックスダンパの開度を変化させてヒート吹出口からの吹出温度を高くする方法が考えられるが、その様な方法では上記熱環境に基づいて決定された必要吹出温度よりも高い温度の空気を吹出させることとなり、車室内の温度が目標室内温度よりも高くなってしまうという問題が生じる。

【0006】そこで、他の方法として吹出モードを変更する方法が考えられる。即ち、上記のような従来の空調自動制御においては、通常、吹出モードとしてベントモード、バイレベルモード（ヒート吹出口とベント吹出口とから吹出すモード）及びヒートモードの中からいずれかのモードを選択するように構成されており、また、バイレベルモードにおいては、所定の必要吹出温度つまり所定のエアミックスダンパ開度の下におけるヒート吹出口からの吹出温度とベント吹出口からの吹出温度とは通常異なり（但し、両吹出口から吹出される空気を合流させた場合の温度は上記所定の必要吹出温度になる）、以下に説明する図1、図2に示すように、エアミックスダンパの開度が同一の場合ヒート吹出口からの吹出温度は上記ヒートモードよりもバイレベルモードの方が高くなるように設定されている。従って、上記のような状況においては、ヒートモードではなくバイレベルモードを選択してヒート吹出口からより高温の空気を吹出させる方法も考えられるが、バイレベルモードにおいては上記のようにヒート吹出口からの吹出温度はヒートモードよりも高くなる反面、ベント吹出口からの吹出温度はヒートモードのときの吹出温度よりも低くなり、従ってベント吹出口から乗員の顔面に向けて冷たい風が吹出し好ましくないという問題がある。

【0007】本発明の目的は、上記事情に鑑み、上記中間季の夜間等において空調が涼しく感じるという問題を回避し、良好な空調を実現できる車両用空調装置を提供することにある。

【0008】

【課題を解決するための手段】本発明にかかる車両用空調装置は、上記目的を達成するために、車室内外の熱環境に基づいて必要吹出温度を求め、該必要吹出温度に基づいて吹出モード及び吹出温度を自動制御する制御手段を備えて成る車両用空調装置であって、上記吹出モードとして少なくともヒートモードとバイレベルモードとヒートデフモードとが設定されると共に、同一の必要吹出温度においてはヒート吹出口からの吹出温度は上記ヒートモードよりも上記ヒートデフモードの方が高く、それよりも更にバイレベルモードの方が高く設定され、ヒートモードにおけるヒート吹出口からの吹出温度よりも上記ヒートデフモードにおけるデフロスト吹出口からの吹出温度のほうが低く、それよりも更に上記バイレベルモ

ードにおけるベント吹出口からの吹出温度の方が低くなるように設定され、上記制御手段は、上記必要吹出温度がヒートモード要求温度範囲の内の低温側の所定温度範囲内でありかつ日射量が所定値より少ない極少の場合は、上記ヒートデフモードに移行させるものであることを特徴とする。

【0009】

【発明の効果】一般に、ヒートモードとヒートデフモードとバイレベルモードとが設定されている場合、上記の様に、同一の必要吹出温度の下では、ヒート吹出口からの吹出温度はヒートモードよりもヒートデフモードのほうが高く、それよりも更にバイレベルモードの方が高くなると共に、上記ヒートモードにおけるヒート吹出口からの吹出温度よりも上記ヒートデフモードにおけるデフロスト吹出口からの吹出温度のほうが低く、それよりも更に上記バイレベルモードにおけるベント吹出口からの吹出温度の方が低くなるように設定される。

【0010】上記車両用空調装置においては、自動制御により選択し得る吹出モードとして上記の如き吹出温度特性を有するヒートデフモードを備え、上記中間季の夜間等に相当する場合、つまり上記必要吹出温度がヒートモード要求温度範囲の内の低温側の所定温度範囲内でありかつ日射量が所定値より少ない極少の場合には、本来ならヒートモードとすべきところを上記ヒートデフモードに移行させることにより、ヒートモードの場合よりもヒート吹出口からの吹出温度（暖房温度）が高く、またバイレベルモードの場合のベント吹出口からの吹出温度よりも高い温度の空気をデフロスト吹出口から吹出させることができ、従ってバイレベルモードの場合のような顔面における冷たさを感じさせることなくヒートモードの場合よりも暖かく感じさせることができ、極めて良好な空調を実現することができる。

【0011】

【実施例】以下、図面を参照しながら本発明の実施例について詳細に説明する。図1は本発明の実施例を示す全体構成図である。

【0012】図に示す車両用空調装置は、ブロア2を備えたブロアユニット4と、エバポレータ6を備えたクーラユニット8と、ヒータコア10を備えたヒータユニット12とを有する空調ユニット14を備えて成る。

【0013】上記ブロアユニット4には内気導入口16と外気導入口18とが形成され、また該内気導入口16と外気導入口18の一方を開放し他方を閉鎖する内外気切換ダンパ20と、該切換ダンパ20を切換作動させるアクチュエータ22とが設けられている。

【0014】上記ヒータユニット12にはヒート送出口24、デフロスト送出口26およびベント送出口28が形成され、各送出口24，26，28はそれぞれ図示しないダクトを介して車室内に向けて開口せしめられたヒート吹出口、デフロスト吹出口およびベント吹出口に接続されている。

【0015】上記ヒート送出口24、デフロスト送出口26およびベント送出口28のそれぞれの近傍には各送出口24，26，28を開閉するためのヒートモードダンパ30、デフロストモードダンパ32およびベントモードダンパ34が設けられ、各モードダンパ30，32，34はヒータユニット12に設けられたアクチュエータ36により夫々連動して開度調節される。

【0016】また、上記ヒータユニット12にはヒータコア10の上流側の通路に位置せしめられたエアミックスメインダンパ38と、ヒータコア10の下流側の通路に位置せしめられたエアミックスサブダンパ40とが設けられ、これらの両ダンパ38，40はヒータユニット12に設けられたアクチュエータ42により夫々連動して開度調節される。

【0017】上記エバポレータ6とヒータコア10とはそれぞれ公知の冷却用熱交換器および加熱用熱交換器である。

【0018】上記空調ユニット14においては、ブロアユニット4におけるブロア2の作動により内気（車室内の空気）もしくは外気（車外の空気）が吸引され、該吸引された空気はクーラユニット8を経由してヒータユニット12に送出され、クーラユニット8およびヒータユニット12においてエバポレータ6およびヒータコア10により適宜冷却および加熱され、しかる後ヒータユニット12の所定の送出口24，26，28から送出され、図示しないダクトを介して上記車室内に向けて開口した所定の吹出口から車室内に吹き出される。

【0019】上記空調ユニット14は制御手段50によって制御される。つまり、制御手段50によって、空気導入モード、吹出モード、吹出温度、送風量等が制御される。上記制御手段50には、上記空調ユニット14を制御するため、外気温（車外の気温）を検出する外気温センサ52、内気温（車室内の気温）検出する内気温センサ54、車室内の日射量を検出する日射量センサ56および空調操作部60からの出力が入力せしめられる。

【0020】上記空調操作部60には、空調装置を作動および停止させるための起動ボタン62と、吹出モードを設定するための吹出モード設定ボタン64と、風量を設定するための風量設定ボタン66と、空気導入モードを内気導入モードと外気導入モードとの間で切換えるための内外気切換ボタン68と、自動制御モードにするための自動モードボタン70と、目標室内温度を設定するための温度設定スライダ72とが設けられている。

【0021】上記吹出モード選択ボタン64は、図示の如く、ベントモード（ベント吹出口のみから吹出させるモード）を設定するベントモード設定ボタンVと、バイレベルモード（ベント吹出口とヒート吹出口のみから吹出させるモード）を設定するバイレベルモード設定ボタンB／Lと、ヒートモード（ヒート吹出口のみから吹出させるモード）を設定するヒートモード設定ボタンHと、

5

デフヒートモード（デフロスト吹出口とヒート吹出口のみから吹出させるモード）を設定するデフヒートモード設定ボタンD／Hと、デフロストモード（デフロスト吹出口のみから吹出させるモード）を設定するデフロストモード設定ボタンDとから成る。

【００２２】上記風量設定ボタン56は、小風量を設定する小風量設定ボタンL_oと、中風量のうちの少ない風量を設定する中小風量設定ボタンM_Lと、中風量のうちの大きい風量を設定する中大風量設定ボタンM_Hと、大風量を設定する大風量設定ボタンHiとから成り、それらのボタンによって４段階の風量設定が可能に構成されている。

【００２３】上記温度設定スライダ72は図中左右方向にスライド可能であり、該スライダ72をスライドさせて左右方向の所定位置に位置させることにより該所定位置に対応する温度（図中左端が最低温であり、右側に行くに従って温度は高くなり、右端が最高温となっている）が目標室内温度として設定される。

【００２４】上記制御手段50による制御は、手動制御と自動制御の双方が可能である。自動制御は、上記自動モードボタン70によって自動制御モードが設定された場合に行なわれる。かかる自動制御は、上記各センサ52，54，56から入力された外気温、内気温、日射量および上記温度設定スライダ72によって設定された目標室内温度に基づいて吹出モード、風量、空気導入モードおよび吹出温度を設定し、それらに応じて下記の様に上記空調ユニット14を制御する。

【００２５】即ち、上記吹出モードについては、上記設定された所定の吹出モードに従って上記アクチュエータ36によりリンク機構を介して各モードダンパ30，32，34が適宜開閉され、上記設定された所定の吹出モードに対応した送出口24，26，28のみが開成される。

【００２６】上記風量については、上記設定された所定の風量に対応した電圧（目標ブロア電圧）が上記ブロア２に供給され、該ブロア２がその電圧に従って回転することにより上記設定された所定の風量に応じた空気が吸引導入されて送出される。

【００２７】上記空気導入モードについては、上記設定された空気導入モードが内気導入モードのときは上記アクチュエータ22により上記内外気切換ダンパ20が外気導入口閉成位置とされて内気導入口16から内気のみが吸引導入され、外気導入モードのときは内外気切換ダンパ20が内気導入口閉成位置とされて外気導入口18から外気のみが吸引導入される。なお、本実施例における自動制御では、急速冷房時以外は外気導入モードとされる。

【００２８】上記吹出温度については、車室内の温度が上記目標室内温度になるように吹出し空気温度の調整が行なわれる。該吹出し空気温度の調整は、上記エバポレータ６に接続された図示しないコンプレッサのオン・オフおよび上記ヒータコア10を通る空気と通らない空気との割合を変化させて空気の温度調節を行なうエアミックスメインダンパ38およびエアミックスサブダンパ40の開度を適宜フィードバック制御することにより行なわれる。

6

【００２９】手動制御は、上記自動モードボタン70によって自動制御モードが設定されていない場合に行なわれ、この場合は、上記吹出モード、風量、空気導入モードがそれぞれ上記吹出モード設定ボタン64、風量設定ボタン66、内外気切換ボタン68によって設定され、それらのボタン64，66，68によって設定された所定の吹出モード、風量および空気導入モードとなるように上記空調ユニット14が上記自動制御の場合と同様にして上記制御手段50により制御される。

【００３０】吹出温度については、手動制御の場合においても、内気温、外気温および温度設定スライダ72によって設定された目標室内温度に基づいて上記自動制御の場合と同様の制御が行なわれる。

【００３１】なお上記空調装置においては、自動制御中に上記吹出モード設定ボタン64、風量設定ボタン66および内外気切換ボタン68によって吹出モード、風量および空気導入モードの手動制御が行なわれた場合、手動制御が優先し、手動によって設定された吹出モード等になるように制御される。

【００３２】次に、上記空調装置における吹出温度と吹出モードの自動制御について更に詳しく説明する。

【００３３】上記空調装置は必要吹出温度検出手段80を備えている。該必要吹出温度検出手段80は、上記各センサ52，54，56および温度設定スライダ72と制御手段50中の必要吹出温度演算部82とから成り、該演算部82により外気温、内気温、日射量、目標室内温度等に基づいて所定の演算式に従い必要吹出温度が演算される。

【００３４】上記制御手段50は、上記必要吹出温度に基づいて吹出温度が該必要吹出温度になるように上記エアミックスメインダンパ38およびエアミックスサブダンパ40の開度を制御する。

【００３５】また、上記制御手段50は、自動制御によって選択し得る吹出モードとして、上記ヒートモード、バイレベルモード、ベントモードに加えてヒートデフモードが設定され、上記必要吹出温度に基づいて予め定められた吹出モードになるように上記アクチュエータ36を介して各モードダンパ30，32，34を開閉制御する。

【００３６】また、上記ヒートモード、バイレベルモードおよびヒートデフモードにおける吹出温度特性は、図２、図３および図４に示すように設定されている。即ち、同一の必要吹出温度つまり同一のエアミックスドア開度においては、ヒート吹出口からの吹出温度は上記ヒートモードよりも上記ヒートデフモードの方が高く、それよりも更にバイレベルモードの方が高く設定されると共に、ヒートモードにおけるヒート吹出口からの吹出温度よりも上記ヒートデフモードにおけるデフロスト吹出

口からの吹出温度のほうが低く、それよりも更に上記バイレベルモードにおけるベント吹出口からの吹出温度の方が低くなるように設定されている。かかる設定は、本実施例では、ベント送出口28とデフロスト送出口26とをヒータコア10を通らない通路側に設けると共にベント送出口28の方をデフロスト送出口26よりもヒータコア10を通った空気が届きにくい上流側に設けることによって行われている。

【００３７】上記必要吹出温度に基づく吹出モードの選択は次ぎの様にして行われる。即ち、上記制御手段50は、図５に示す様に、上記各センサ52,54,56および空調操作部60から入力された外気温、内気温、日射量及び目標室内温度に基づいて必要吹出温度を計算し、該必要吹出温度が低のときはベントモード、中のときはバイレベルモード、高のときはヒートモードを選択する。

【００３８】上記必要吹出温度に基づいた吹出モードの選択を、図６および図７を参照しながら説明する。即ち、図７に示すように、S1において必要吹出温度T_FがA（＋方向の変化で25℃未満）かB（－方向の変化で20℃より大）かを判断する。Aのときは S8に進み、フラグFLを０にリセットし、S9でベントモードを選択する。Bのときは S2に進み、フラグFLが１か否かを判断する。フラグFLが１でない場合（フラグFLが１のときは以下のS14,15で説明するようにヒートデフモードになっているとき、フラグFLが１でないときはヒートデフモード以外のとき）は S3に進み、そこで必要吹出温度T_FがA（＋方向の変化で35℃未満）かB（－方向の変化で30℃より大）かを判断する。Aのときは S10 に進み、フラグFLを０にリセットし、S11 でバイレベルモードを選択する。Bのときは S4に進み、必要吹出温度T_Fが32℃より小か否かを判断する。32℃以上のときは S12 に進み、フラグFLを０にリセットし、S13 でヒートモードを選択する。32℃より小のときは S5に進みそこで日射量が所定値I_S（I_Sは０に近い微少値）より小か否かを判断し、I_S以上のときは S12 に進み、フラグFLを０にリセットし、S13 でヒートモードを選択する。I_Sより小の微少量のときは S14 に進み、フラグFLを１にセットし、S15 でヒートデフモードを選択する。

【００３９】一方、上記S2でフラグFLが１であった場合は、S6に進み、必要吹出温度T_Fが29℃より大か否かを判断する。29℃以下のときは S10 に進み、フラグFLを０にリセットし、S11 でバイレベルモードを選択する。29℃より大のときは S7に進み、必要吹出温度T_Fが35℃より小か否かを判断する。35℃以上のときは S12 に進み、フラグFLを０にリセットし、S13 でヒートモードを選択する。35℃より小のときは S14,S15 に進み、フラグFL＝１およびヒートデフモードを維持する。以上説明した吹出モードの選択を図６においてグラフで示す。

【００４０】上記の必要吹出温度に基づく吹出モードの選択は、要するに、上記必要吹出温度がヒートモード要求温度範囲即ち－方向の変化で30℃以上範囲のときは本来ならばヒートモードが選択されるのであるが、かかるヒートモード要求温度範囲の内の低温側の所定温度範囲内即ち30℃～32℃のときは、日射量が所定値より少ない極少の場合は、上記ヒートモードからヒートデフモードに移行させるというものである。

【００４１】上記車両用空調装置においては、自動制御により選択し得る吹出モードとして上記の如き温度特性を有するヒートデフモードを備え、上記中間季の夜間等に相当する場合、つまり上記必要吹出温度がヒートモード要求温度範囲の内の低温側の所定温度範囲内でありかつ日射量が所定値より少ない極少の場合には、本来ならヒートモードとすべきところを上記ヒートデフモードに移行させることにより、ヒートモードの場合よりもヒート吹出口からの吹出温度（暖房温度）が高く、またバイレベルモードの場合のベント吹出口からの吹出温度よりも高い温度の空気をデフロスト吹出口から吹出させることができ、従ってバイレベルモードの場合のような顔面における冷たさを感じさせることなくヒートモードの場合よりも暖かく感じさせることができ、極めて良好な空調を実現することができる。

【００４２】なお、上記実施例においては、一旦ヒートデフモードに移行した後は必要吹出温度が29℃以下になったときにバイレベルモードに移行し、35℃以上になったときにヒートモードに移行するように設定されているが、このヒートデフモードから他の吹出モードへの移行条件は適宜に設定すれば良いものである。また、上記必要吹出温度は、必要吹出温度そのものである必要はなく、必要吹出温度に相当するものであればなんでも良い。

【図面の簡単な説明】

【図１】本発明の実施例に係る車両用空調装置を示す側面図

【図２】ヒートモードの吹出温度特性を示す図

【図３】バイレベルモードの吹出温度特性を示す図

【図４】ヒートデフモードの吹出温度特性を示す図

【図５】吹出モードの選択手順の概略を示すフローチャート

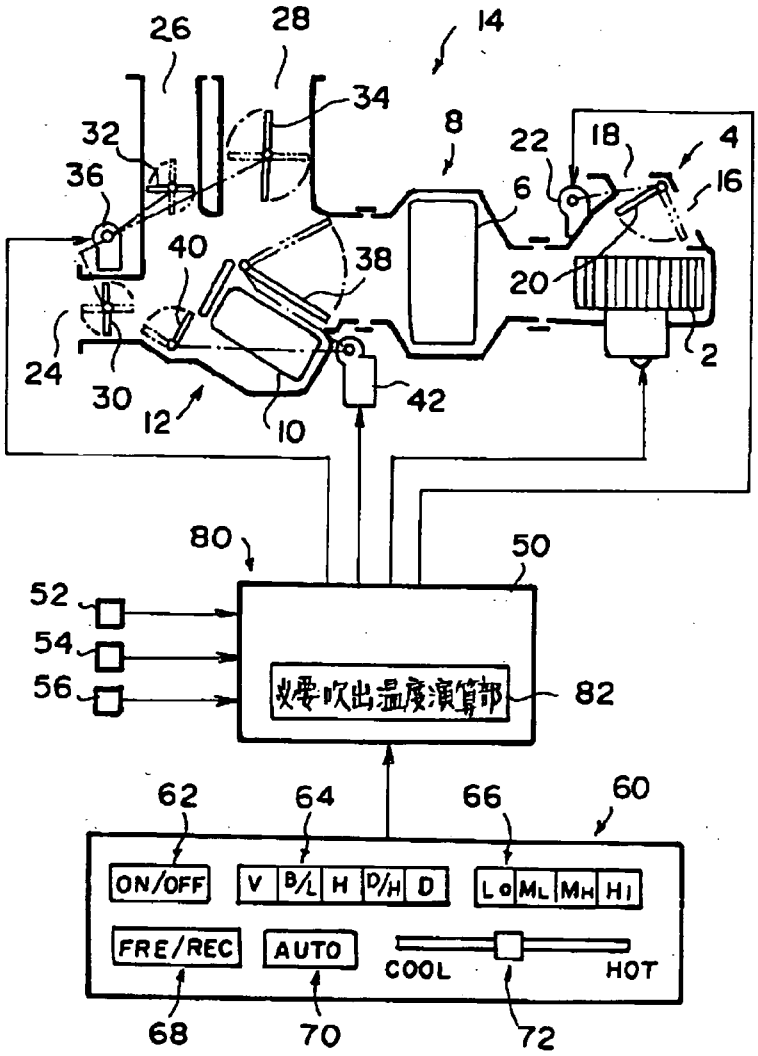
【図６】吹出モードの選択態様を示すグラフ

【図７】吹出モードの選択手順の詳細を示すフローチャート

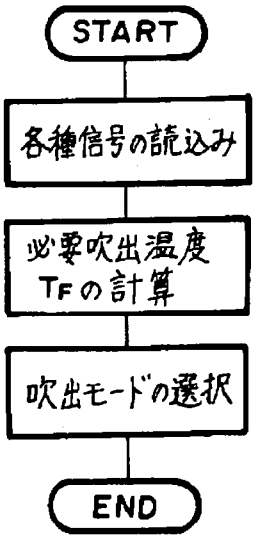
【符号の説明】

50　制御手段

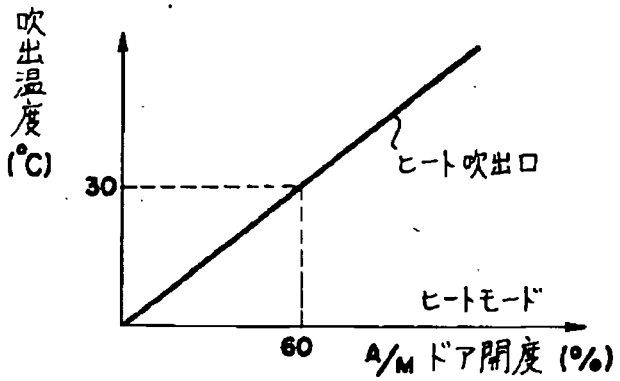
【図1】


26
28
14
34
8
22
18
4
16
6
32
36
40
38
20
24
30
12
10
42
2
80
50
52
54
56
必要吹出温度演算部
82
62
64
66
60
ON/OFF
V
B/L
H
D/H
D
Lo
ML
MH
HI
FRE/REC
AUTO
COOL
HOT
68
70
72

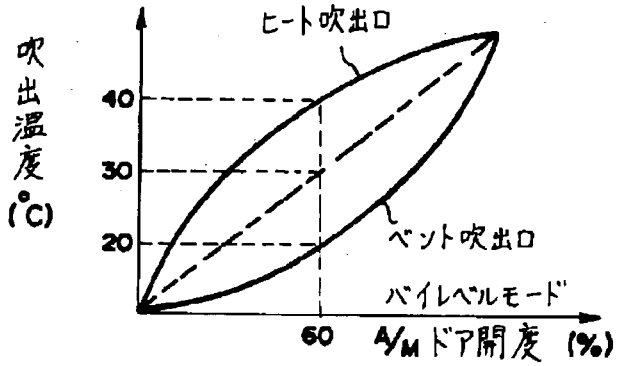
【図5】


START
各種信号の読込み
必要吹出温度
TFの計算
吹出モードの選択
END

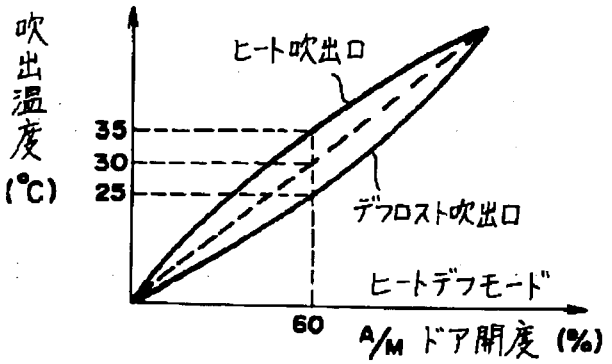
【図2】


吹出温度
(°C)
30
ヒート吹出口
ヒートモード
60
A/M ドア開度 (%)

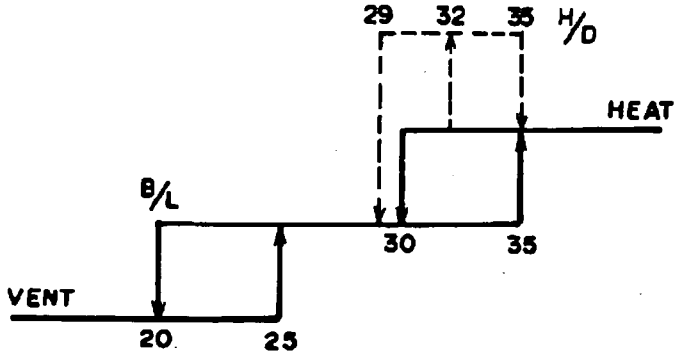
【図3】


吹出温度 (℃)
ヒート吹出口
40
30
20
ベント吹出口
バイレベルモード
60
A/M ドア開度 (%)

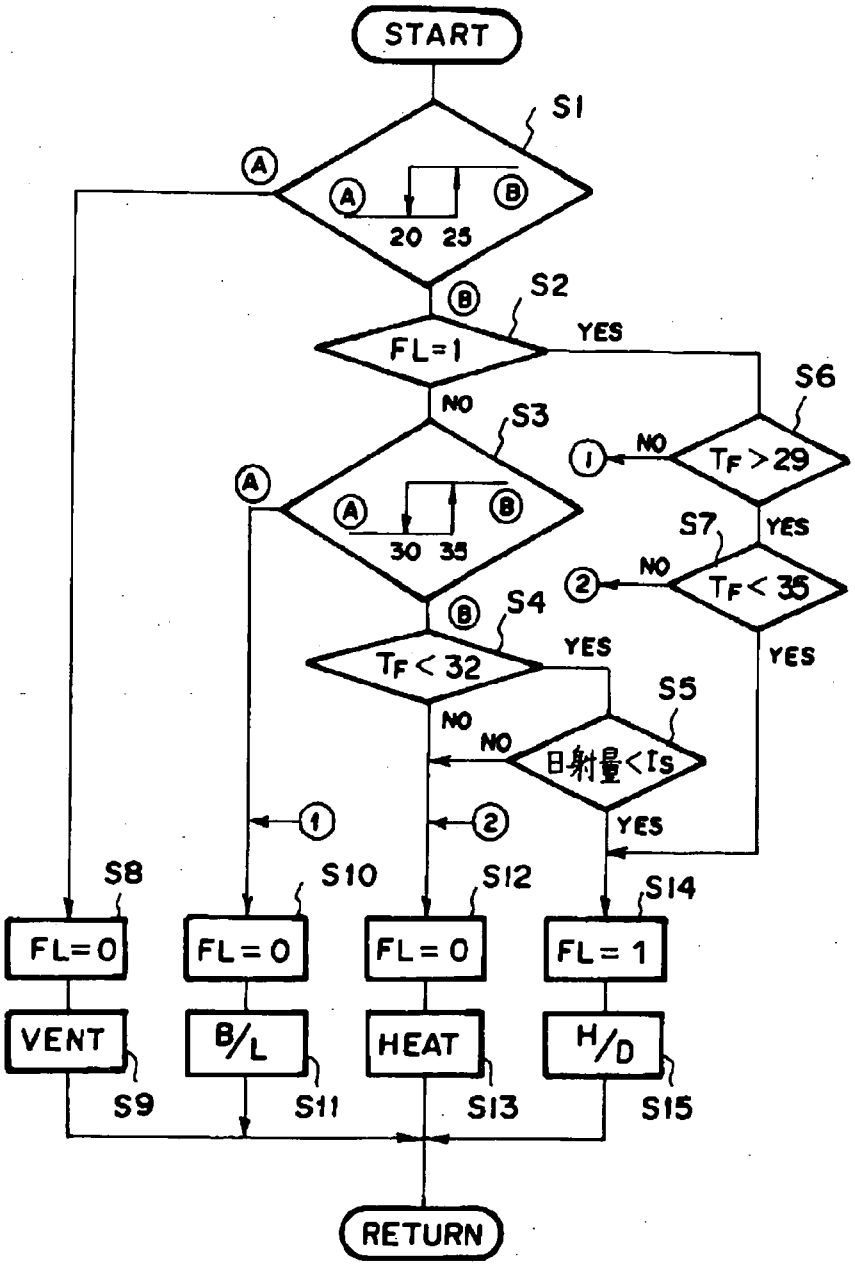
【図4】


吹出温度 (℃)
ヒート吹出口
35
30
25
デフロスト吹出口
ヒートデフモード
60
A/M ドア開度 (%)

【図6】


29
32
35
H/D
HEAT
B/L
30
35
VENT
20
25

【図7】


START
S1
A
A
B
20 25
B
S2
FL=1
YES
NO
S6
S3
NO
$T_F > 29$
A
A
B
30 35
S7
YES
NO
$T_F < 35$
B
S4
YES
$T_F < 32$
S5
NO
NO
日射量<Is
YES
1
2
S8
S10
S12
S14
FL=0
FL=0
FL=0
FL=1
VENT
B/L
HEAT
H/D
S9
S11
S13
S15
RETURN

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 649 121 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94307596.0**

(22) Date of filing : **17.10.94**

(51) Int. Cl.[6] : **G07F 17/16**, G06F 17/60, G06F 17/30

(30) Priority : **15.10.93 US 137880**

(43) Date of publication of application : **19.04.95 Bulletin 95/16**

(84) Designated Contracting States : **AT BE CH DE ES FR GB IT LI NL SE**

(71) Applicant : **International Business Machines Corporation**
Old Orchard Road
Armonk, N.Y. 10504 (US)

(72) Inventor : **Tsevdos, James T.**
2711 NE 57th Street
Fort Lauderdale, Florida 33308 (US)
Inventor : **Cook, Ross L.**
901 SW 36th Avenue
Boynton Beach, Florida 33435 (US)
Inventor : **Ring, Nancy Lee**
6129 Town Colony Drive
Boca Raton, Florida 33433 (US)
Inventor : **Barnhill, Robert S.**
21218 St. Andrews Blvd.,
No. 10-405
Boca Raton, Florida 33433 (US)
Inventor : **Hamblin, Glen E.**
9384 Gettysburg Road
Boca Raton, Florida 33434 (US)
Inventor : **Milstead, Kenneth L.**
9927 Majestic Way
Boynton Beach, Florida 33437 (US)
Inventor : **Kindell, Craig N.**
2014 SW 29th Court No. 2A
Delray Beach, Florida 33445 (US)
Inventor : **Waefler, Susan Elizabeth**
5086 Madison Road
Delray Beach, Florida, 33484 (US)
Inventor : **Portela, Carlos**
2659 SE 14th Street
Pompano Beach, Florida 33062 (US)
Inventor : **Anderson, Brent C.**
14232 Marsh Lane
Dallas, Texas 75244 (US)

(74) Representative : **Burt, Roger James, Dr.**
IBM United Kingdom Limited
Intellectual Property Department
Hursley Park
Winchester Hampshire SO21 2JN (GB)

(54) Digital information accessing, delivery, and reproduction.

(57) A digital data on-demand turnkey system at a customer premise wherein N number of servers provide for 100% of content distribution of remotely stored digitized information, which information may be previewed in real-time, and product incorporating selected digitized information can be manufactured on-site and within a short response time to a customer's request at a point of sale location. In a retail environment customers, at a point-of-sale location are able to exhaustively search and preview the content database using graphics-based touch screens at consumer kiosks. Previews including audio and video segments are made available. Prompting screens allow customers to make purchasing decisions by stipulating content which is available from any number of categories of subject matter including music. Content can be identified by album, artist or partial portions of material within the content of the totality of the work. The selected media for the manufacture and production of the digital data may be from a myriad of different selections and can include CD's, cassette tapes, CD ROM technology, reel-to-reel tapes, and video disks, as an example. Since the system is adaptable to having an unlimited number of servers, strategic and economic networking is accomplished to allocate proportionally the preview material as well as the total content of a specific work at the respective servers depending, in particular, on the popularity of the content and the costs of transmission. A master server will be situated geographically so as to be accessible to chain and network subservers. Its geographic and networked location is dependent upon communication network systems and subsystem costs and availability in order to best serve a customer's premise, whether it be a retail store or similar point-of-sale or other end-user location. The strategy of allocating the content between the servers, preview and for the final manufacturing process for a particular work, is done so as to eliminate the need for any in-store inventory of, for example, an artist's composition regardless of format. Security mechanisms that require centralized database authorizations prior to the transmission of content and/or the manufacture of any of the products is provided in addition. The systems provides simplified financial and accounting services. The total authorization, accounting and management reporting and

EP 0 649 121 A2

marketing system collects and maintains data useful to the retailer and the content data owners or managers, such as label companies.

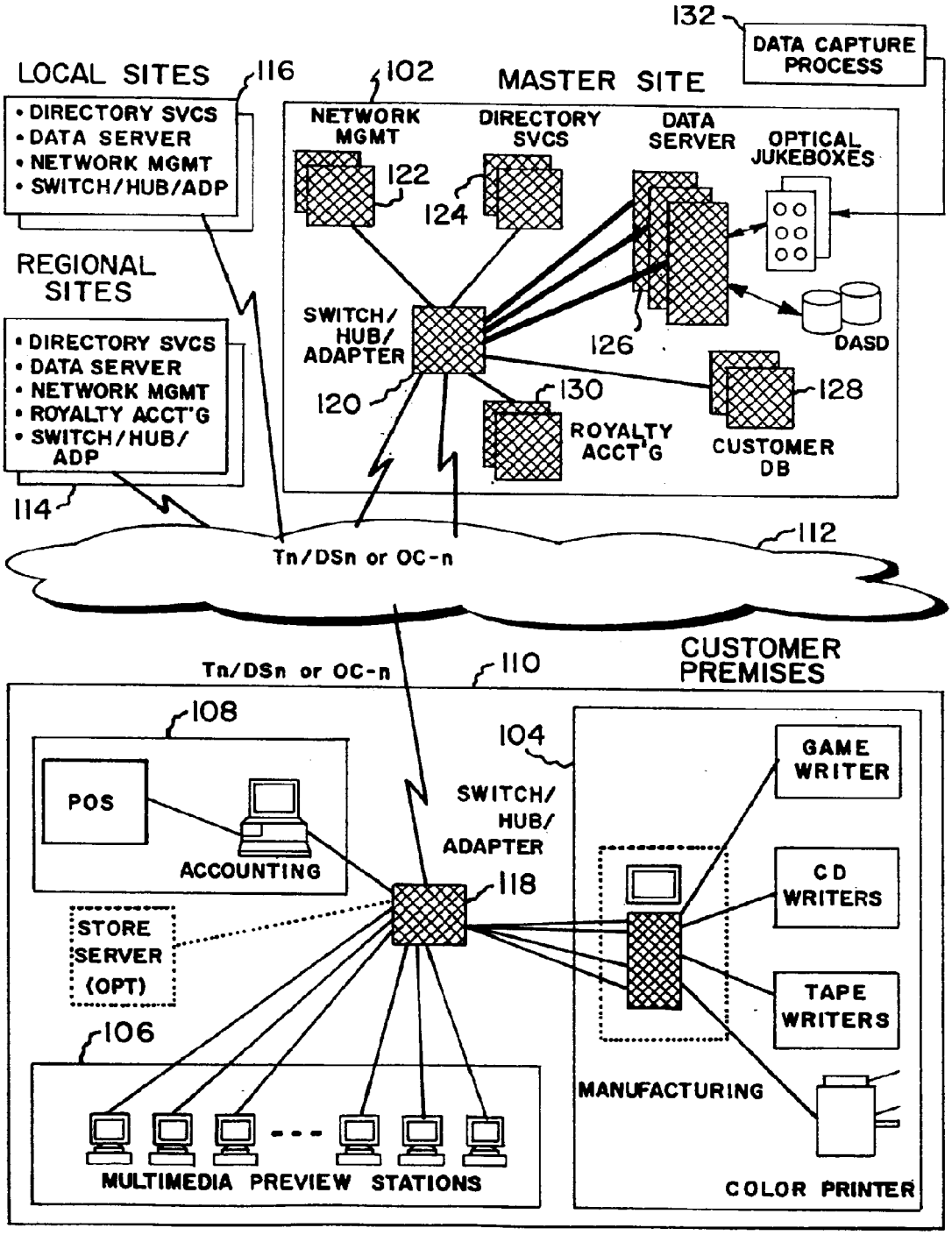


FIG. 1

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to the field of data storage and data communication. In particular, the invention relates to a system for accessing digital information, such as audio and video information, at remotely stored locations and for communicating in real-time that information to another location (such as a user's premise or a point-of-sale) for reproduction of the communicated information in a material object for end-user use.

Description of Related Art

Retailers of the world are in a constant struggle to meet the demands of the consuming public with regard to the availability and competitive pricing of products. This struggle is a challenge for the retailer's ability to meet the end-user's immediate need and if an item must be backordered, the customer's patience. The retailer must do that in a manner which is cost effective to the merchandiser and competitive from a delivery and price standpoint to the purchaser.

There is likely no more of a demanding and aggressive purchasing public than those individuals who are avid followers and fans of entertainers and artists in the entertainment industry. Because of the high demand and the massive number of customers and potential customers, as it relates to the entertainment industry, the level of competition is extremely high, whether it be for memorabilia, contemporary news items, fashion trends, or entertainment products.

Product distribution and availability is one of those merchandising areas where the retailer has experienced keen competition and where they have struggled to be the source of all products, to be identified as having immediate availability, to continue to serve the totality of the market share which they have captured and to grow that market share.

As an example, in the entertainment industry, particularly as it applies to companies involved in music typically known as music label companies, there has been a constant challenge for such companies to exploit their entire catalog. There is a need or concern to make every album in their catalog available in any format and to never be out of stock with respect to that particular product. If the labels are indeed able to achieve that high level of expectation and availability, the retailer then is faced with the dilemma of allocating floor space to all of that available inventory regardless of the popularity of much of the stocked material. If this is not common practice, the retailer will be faced with losing a particular customer and clientele if the stock is not on-hand. In most instances, it is unrealistic to expect that a retailer will have sufficient floor space and financial resources to have an exhaustive stock of all products. As an example, the music industry is presently offering 60,000 to 100,000 albums by title and is growing at an extraordinary rate. Additionally, artists are continually engaged to expand their repertoire. This is particularly true when an artist has gained recent popularity and is a big hit with the consuming public, which may have resulted in an insatiable appetite for that artist's work. The retailer has the challenge of identifying what quantities are marketable over what period of time for that artist and any number of other artists and labels. The outlet merchandiser is fraught with the effort to stock those products and to have them available when approached by a customer who has that immediate and demanding interest in procuring a copy of that artist's work. Any miscalculation by the retailer evolves in what is known as stock-outs.

The retailer's inventory management problems are there whether or not the product is in inventory at their particular retail outlet or needs to be procured in order to meet that immediate demand regardless of the life of the demand. These are time consuming and difficult calculations for retailers. The retailer break-even point is in a constant state of flux and miscalculations and bad judgment calls result in prices which are less than competitive and margins which are often nonexistent.

A further frequently encountered problem by the retailer is the issue of a purchaser who may not be in the position to fully identify either the artist by full name, the label company or the full name of the title of the authored work or piece. Now with respect to items which are of current popularity, some developed systems are available to preview portions that may be able to aid in the identification of the particular product of interest. However, when it comes to the literally hundreds of thousands of pieces that have been produced, it is a near impossibility under the present systems to assist that potential purchaser when that artistic production is not of current vintage and even may be extremely dated. Further, even if some assistance has been given to that particular purchaser, without preview of that item, there is a great deal of uncertainty on the part of the potential purchaser as to whether or not the item identified is one of interest to them. In most cases there is great hesitancy to purchase an item which may be relatively expensive and which may turn out to not be a item of particular interest.

It is difficult for the retailer to maximize the inventory at the retail outlet and have it attain the level of expectation of the customer. Inventory and product stocking principles, such as just-in-time practices, have assisted the merchandiser to control the amount of stock-on-hand against cash-flow concerns to meet customer demand. But these practices still fall short of the inpatient consumer who wants the product now regardless of the reasonableness of the demand, even for products which are so popular that stock-outages are to be expected or products which are so dated that only archival copies generally exist.

Even in today's automated systems which provide a computerized library resource of available programmed materials, limitations are imposed and financial constraints are present due to a lack of convenient and available storage space and constricted and time consuming data communication channels.

SUMMARY OF THE INVENTION

In one aspect, the present invention provides a method for accessing digital information stored in a source library at a first location and recording a portion of the digital information, which information is processed for subsequent playback, onto a recordable medium at a location other than the first location, the method including the steps of: identifying, at a second location remote from the first location, a portion of the digital information in the source library; sending a request from the second location to the source library for the identified portion of the digital information in the source library to be transferred from the source library; validating that the identified portion of the digital information is authorised for playback; transferring in real-time information within the identified portion of the digital information to a manufacturing device at a location other than the first location for.incorporation of the digital information on a medium suitable for playback; and incorporating the identified portion of the digital information on the medium in real-time in a manner suitable for playback.

In a second aspect, the invention provides a method of allocating for storage 100 percentage of content data amongst a plurality of data servers remotely disposed to each other and connected in hierarchal levels having a low and high end and in data communication with a data accessing station also remotely disposed from each of the plurality of data servers, but less so disposed from the data server at the low end of the hierarchy than from each succeeding data server at the next succeeding higher level of the hierarchy through and to the data server at the high end of the hierarchy of data servers, comprising the steps of: allocating for storage X percentage of the content data at the data server at the low end of the hierarchy least remote from the data accessing station which X percentage of allocated content data is a function of content data for most frequent accessing by the data accessing station; allocating for storage Y percentage, where Y is < 100 and $> X$, of the content data at the data server at the high end of the hierarchy of data servers most remote from the data accessing station as a function of content data for least frequent accessing by the data accessing station; and allocating for storage 100-X-Y percentage, of content data between any other data servers disposed between the low end and the high end data servers, allocating for each succeeding data server more remote from the data accessing station a greater percentage of content data for storage than an allocation for any data server less remotely disposed from the data accessing station and a lesser percentage of content data than the next succeeding server more remotely disposed from the data accessing station.

In a third aspect, the invention provides a data communication system for transmitting digital data between remotely located system stations where each station is adapted to participate in a process of storing, accounting for and manufacturing media with digitized content data packaged for playback, comprising: a point of access station for identifying selected digitized content data and for generating a communication requesting said content data; a content data storage station having digitized content data remotely disposed from the point of access station; an accounting station for recording transactional data for each customer point of access station communication requesting digitized content data from the content data storage station; a manufacturing station remotely disposed from said content data storage station for generating media incorporating the digitized content data requested by the point of access station from the content data storage station; and a communication network for interconnecting the remotely disposed content data storage station with the point of access station, the accounting station and the manufacturing station to transfer digitized content data in real-time between the content data storage station and the manufacturing station.

In a fourth aspect, the invention provides a multimedia preview station including an audio system, a touchscreen monitor and a controller for individually presenting a plurality of screens on the monitor for touch control by a consumer to present to the consumer consumer identified and requested screen presentations, comprising; a library screen identifying a plurality of categories of content data for screen presentation on the monitor including a touch point for activating the presentation of a category; an alphabetical listing screen identifying content data for presentation by alphabetical listing of the category subject matter of the library screen including a touch point for activation by the consumer; a search screen including monitor touch points for identifying a consumer request either alphabetically or numerically; a preview screen including monitor touch points

for previewing consumer identified audio and video listed content data transferred to the preview screen in real-time and monitor touch points for selecting for purchase listed content data; and a purchase selection screen including purchase selections made by the consumer, listed by item selected, quantity, format and price, including monitor touch points for altering the item selection.

In a fifth aspect, the invention provides an accounting system including a touchscreen monitor and a controller for individually presenting a plurality of screens on the monitor for touch control by an operator to present to the operator, accounting worktools for analysis of remotely stored digitized data of consumer purchases as a function of sales operation, quantity, monetary amount, purchase format, purchase subject matter and consumer demographics, comprising: a report screen identifying a plurality of analysis reports for screen presentation on the monitor including a touchpoint for activating the presentation of an analysis report as a function of consumer sales, consumer demographics, sales unit, and sales group; a digitized data storage unit remotely disposed from the report screen for transferring at a high speed constant bit rate the digitized data corresponding to a touchpoint activated analysis report on the report screen of the touchscreen monitor; and an analysis report screen presenting in quantified format the remotely stored digital data corresponding to consumer purchases.

In a sixth aspect, the invention provides a manufacturing system for manufacturing media with digital data content including a plurality of data servers storing digital data content remotely disposed to each other and connected in hierarchal levels comprising: top level servers each storing a high percentage of the stored digital data content; intermediate level servers each storing a lesser percentage of stored digital data content than the top level servers; a high level communication network for interconnecting the top level servers and the intermediate level servers and for transferring digital data content between the top level servers and the intermediate level servers in asynchronous transfer mode; end site servers each storing a lesser percentage of stored digital data content than the intermediate level servers; a low level communication network for interconnecting the intermediate level servers and the end site servers and for transferring digital data content between the intermediate level servers and the end site servers in asynchronous transfer mode; and a manufacturing device interconnected to one of the end site servers for receiving in asynchronous transfer mode digital data content for manufacturing media with the received digital data content.

The present invention enables the provision of a digital data on-demand turnkey system wherein N number of servers provide for 100% of content distribution of remotely stored digitized information, which information may be previewed or reviewed in real-time, and products incorporating selected digitized information can be manufactured within a short response time to a customer's request at a point of sale location. This technique is carried out without intermediate buffering or caching which minimizes costs particularly at those locations being served. The Oxford Dictionary of Computing defines a real-time system as any system in which the time at which output is produced is significant. It states that this is usually because the input corresponds to some movement in the physical world, and the output has to relate to that same movement. It notes that the lag from input time to output time must be sufficiently small for acceptable timeliness.

It is preferred for consumers in a retail environment, whether they be at a point-of-sale or at another customer premise, to be able to exhaustively search and preview the content database using graphics-based touch screens at consumer interface terminals such as a kiosk. For a proposed audio application, previews including audio and video segments are available. Through the use of prompting screens customers are able to make purchasing decisions by stipulating content which is available from any number of categories of subject matter such as music types. Content can be identified by album, artist or remembered portions of material within the content of the totality of the work.

The selected media for the manufacture and production of digital and analog software may be from a myriad of different selections and can be accomplished on digital media such as CD's, analog technology such as cassette tapes, CD ROM technology, reel-to-reel tapes, game cartridges, video disks of varying sizes, sheet music, floppy disks and can include content which is audio, such as voice or music, video, interactive game program and cartography, just to mention a few.

Since the system is adaptable to having an unlimited number of servers, strategic and economic networking can be accomplished to allocate proportionally the preview material as well as the total content of a specific work at the respective servers depending, in particular, on the popularity and the costs. A master server will be situated geographically so as to be accessible to chained and networked subservers. Its geographic and networked location is dependent upon communication network systems and subsystems costs and availability in order to best serve a customer's premise, whether it be a retail store or similar point-of-sale or other end-user location. The strategy of allocating between the servers the content for preview and for the final manufacturing process for a particular work, is strategized in order to eliminate the need for any in-store inventory of an artist's composition regardless of format. In essence, every album in every particular label's catalog will be available in any format and the retailer will never have a condition of being out of stock. The retailer will be

able to utilize floor space to maximize merchandising and need not feel the burden of inventory drag.

Thus , merchandising is greatly enhanced through the use of multi-media kiosks or booths individualized to the customer and/or categories of buyers. This makes the selection, preview and purchase more personal to the customer. Not only is the customer able to preview any number of one of a selected categories of music types or whatever the content subject matter may be, including electronic games, software programs, video, audio and images, the customer can be introduced to other albums or works of a particular artist or a particular type of music. With the multi-media kiosk experience, the retailer has an opportunity to further enhance the purchaser's interest in matters which possibly are collateral to the original intent, interest and desire of a customer. Obviously, this is of great advantage to the retailer and can result in additional sales. The gathering and further use of artificial intelligence of the sort which is specific to a buyer or to a category of buyers can maximize the retailer's return through the use of such a system.

Both the retailer and the label companies are and have been specially interested in the ability to protect the myriad of proprietary rights which the artists, the labels and others have through copyright protection and/or licenses. The present inventive system provides for security mechanisms that require centralized database authorizations prior to the transmission of content and/or the manufacture of any of the products. In addition, the system simplifies the financial and accounting aspects and will reduce, if not totally eliminate, pilferage of items within the store at the point of sale.

Further, before a purchase is actually made, numerous communications are made to the potential purchaser ensuring that the purchaser has the properly identified items of interest and is interested in their procurement. These aspects and the further aspects of the preview opportunities greatly reduce and minimize return of purchases by the customer.

The total authorization, accounting and management reporting and marketing research system collects and maintains data useful to the retailer and the label companies. With proper analysis all stations along the merchandising trail can benefit from this information and exploit the economic opportunities available from the analysis of such data.

After review, selection and proper authorization procedures have transpired, all of which are done in a short period of time having benefitted from the expedited communication and transfer of digitized information to provide for real time handling of the customer's needs at the point of sale, the manufacturing and production process of the media in the selected format is accomplished. Not only is the manufacturing subsystem capable of generating an exact replica of the master of the selected piece or work, but the manufacturing subsystem is also capable of producing the art work or labels and any included documentation with regard to the purchased item. The digitized information to be replicated will be communicated to the manufacturing subsystem at the customer premise from any one or a number of the data servers, including the master server within the geographical network of servers. As stated before, the specific server from which the information or a portion of the information is to be forthcoming is dependent upon such factors as the popularity of the work, the total length of the work, the cost of transporting the information from the server to the customer premise and other factors which are cost effective and minimize the distance to the customer premise while maintaining the required level of response for real-time transactions.

In the above mentioned way, the present invention provides an effective means for accessing remotely stored information at strategically located server locations adapted to communicate and transmit content information to a customer premise in real time and in such a manner so as to be able to manufacture media incorporating the transmitted information, while maintaining the integrity and providing accountability of the communicated and networked information to provide on-demand and real time customer satisfaction.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a pictorial representation of a system architecture including a master site having communication with a retail store according to the principles of the present invention;

FIG. 2 is a high-level communication network structure, exemplary of a communications system for implementing the system and method of the present invention;

FIG. 3 is an exemplary remote serving topology of a hierarchical arrangement of the servers with allocated content represented graphically opposite the corresponding hierarchical layer of the servers;

FIG. 4 is an exemplary regional data server illustrating a uniprocessor architecture;

FIG. 5 is an alternative exemplary data server illustrating a multi-processor architecture;

FIG. 6 is an exemplary server architecture illustrating communication interconnecting through a ring or bus cluster;

FIG. 7 is an alternative server architecture illustrating communication interconnecting through a switch cluster arrangement;

FIG. 8 is an exemplary switch for connecting different communication service consumers;
FIG. 9 is an exemplary hub or router for connecting different communication service consumers;
FIG. 10 is an alternative communication connecting system including an adapter and a computer for connecting different communication service consumers;
FIG. 11 is a exemplary block diagram of an on-demand system with illustrative switching devices for networking between regional and central servers, according to the principles of the present invention;
FIG. 12 is an exemplary partial block diagram of a retail store system with expanded illustration of the manufacturing subsystem.
FIG. 13 is an exemplary block diagram of a retail store system according the principles of the present invention with the data server and other subsystems within a local area network (LAN);
FIG. 14 is a functional block diagram of the on-demand data system according to the principles of the present invention;
FIG. 15 is a logic diagram of the on-demand data system illustrating its five major subsystems without regard to local or remote location;
FIG. 16 is an exemplary illustration of the multimedia preview station of the on-demand system according to the principles of the present invention;
FIG. 17 is an exemplary illustration of the multimedia preview station alternatively showing a set-top video decoder application;
FIG. 18 is a monitor screen illustration of the library content of the on-demand audio and video system at the preview station;
FIG. 19 is a multimedia preview station screen presentation of the audio and video music categories at the preview station;
FIG. 20 is a preview station screen presentation of an exemplary listing of artists and groups within a particular music category;
FIG. 21 is a preview station screen illustration of a keyboard layout for manipulation by touch for identifying through description artist's works;
FIG. 22 is an exemplary preview station screen listing of a particular search result;
FIG. 23 is an exemplary preview station screen listing of the album works of a particular artist or group;
FIG. 24 is an exemplary preview station screen listing of selections for preview of a particular album;
FIG. 25 is an exemplary preview station screen depicting a single selection for audio preview;
FIG. 26 is an exemplary video preview station screen with motion video of a selection;
FIG. 27 is an exemplary preview station screen listing and itemization of selected items in the shopping cart of the previewer;
FIG. 28 is an exemplary preview station screen presenting a previewer with the opportunity to submit or cancel an order;
FIG. 29 is a screen illustration of the business analysis tool workstation illustrating the categories for detailed accounting and finance information;
FIG. 30 is an exemplary screen illustration of an accounting for a particular point-of-sale outlet by product format;
FIG. 31 is an exemplary screen illustration of the high volume album selections for a particular time period.
FIG. 32 is an exemplary screen view of an alternative accounting tool illustrating sales activity by album and music category;
FIG. 33 is an exemplary graphical screen presentation of retail activity by type of music;
FIG. 34 is an exemplary graphical screen presentation of retail activity by type of media;
FIG. 35 is an exemplary accounting workstation screen chart showing demographic statistics for sales of product types;
FIG. 36 is an exemplary screen presentation of accounting workstation analysis information which is store-hours time dependent;
FIG. 37 is an exemplary central host accounting workstation screen analysis tool for local, regional and chain accounting information; and
FIG. 38 is an exemplary hardcopy printout report of a central host analysis accounting tool showing sales for the regional central host by music category.

DESCRIPTION OF THE ILLUSTRATIVE EMBODIMENTS

While the present invention will be described more fully hereinafter with reference to the accompanying drawings, in which illustrative embodiments of the present invention are shown, it is to be understood at the outset of the description which follows that persons of skill in the appropriate arts may modify the invention

here described while still achieving the favorable results of this invention. Accordingly, the description which follows is to be understood as being a broad, teaching disclosure directed to persons of skill in the appropriate arts, and not as limiting upon the present invention.

Many of the deficiencies, inadequacies, restrictions, limitations both technically and commercially and the cost prohibitions of the presently available on-demand packaged digital data for playback systems are overcome by the present invention. Presented is a concept where a device for manufacturing media with content data has virtual connection to a host content data machine where all message packets arrive and in sequence to a host content data machine at a high speed constant bit rate, providing for data to be streamed from a Master Site 102 to a Manufacturing subsystem 104 as shown in FIG. 1. In the context of the present invention a data stream is a continuous stream of data elements being transmitted, or intended for transmission, in character or binary-digit form. In a preferred embodiment of such a system a data preview station 106 is also provided and high speed data streaming is a desirable feature of the operation for preview. While it is not necessary to have high speed data streaming for the transaction accounting operations of the Master Site 102 and a Accounting Subsystem 108 at, e.g., the retail store 110, that also is preferred.

In a preferred manufacturing system with accountability, high speed data streaming would be from end station to end station. One end station would be for data preview at booths 106. Another end station is the manufacturing device or subsystem 104. Yet another end station is the manager's or the accounting work station 108. While a further end station is the data content host or the Master Site 102.

An important aspect of streaming data between the end stations is for the data to be streamed so that it can be presented so as to reproduce whatever the recorded information is at the proper reproduction rate so that the individual using it or looking at it or listening to it can not distinguish whether or not the recorded version is being played back from a local media, such as a video tape or compact disc, or if the individual is witnessing it being played back remotely and transmitted through a network. The bits of information have to be delivered to the preview or manufacturing device at a rate required by that device to recreate faithful reproduction of the original recording.

A communications cloud, such as cloud 112 in FIG. 1 generically represents a network where included state-of-the art topology provides for routing and transporting of data. In general, the interface into the cloud is specified by the myriad of different network topologies. So regardless of the specific topology or type of a backbone or switching techniques within the cloud, the data gets transported from one point to another point over a wide geographical area.

The Tn designations of FIG. 1, or T Carrier transmission systems are represented by DSn or Digital Signal levels which make up the standard digital transmission hierarchy developed by Bell Labs. DSO represents a single digital voice channel at 64 Kilobits per second. T 1 represents 24 DSO channels in the US., and in Europe the designation is E 1 representing 30 DSO channels. The different worldwide digital hierarchy's are referenced in their applicable standards body's such as Bellcore Technical References, International Telecommunications Union (CCITT) and International Organization for Standardization. The OC-N designation represents the Sonet Optical Carrier Speed Hierarchy where OC-3 is equivalent to 84 T 1 or 2016 DSO voice channels. The text Data Network Design authored by Darren L. Spohn copyrighted 1993 by publisher McGraw-Hill, Inc. presents design information relied upon in the development of the present invention.

In the illustration of FIG. 1, coming into the customer premise site or Retail Store 110, as for example, either a T1 or DS1, which is 1.5 megabits or a T3 or DS3 level which is 44 megabits per second or an asynchronous transfer mode (ATM) which would be carried over in OC-3 optical fiber which provides 155 megabit fiber connection or a combination is employed. ATM is a cell relay type of protocol which is being widely developed throughout the telephony industry. At the other side of the exemplary cloud 112 where a serving site exists and, in the illustration of FIG. 1, Master Site 102, there is a exemplary connect to T3, DS3 which follows with ATM running over either OC-3 fiber at 155 megabit or even up to OC-12 which is 622 megabits.

At the top level server site, the actual bandwidth or bit rate required would be dependent on the number of, and the size and the quantity of the data that is served from that site. Multiple T3s or multiple OC-3s or OC-12s are connected and multiplexed out to the different devices in the Master Site 102. This is also the practice for any of the server sites, such as Regional Site 114 and Local Site 116, as well as the top level sites or Master Site 102.

FIGS. 8, 9, 10 illustrated three approaches to multiplex and demultiplex the aggregate bandwidth required by the various terminals and devices contained within the customer premises to a single high speed trunk entering the network. This approach is also applied at the server sites to aggregate bandwidth entering the network. Another important function of the switches 118 or 120 in FIG. 1, and in FIGS. 8, 9, 10, is to multiplex or aggregate multiple trunks from the network provider to the customer premise or server site.

The ATM communication technique presents a standard 53 octet or 53 byte cell which has included within it a header comprising the virtual path identifier, the generic flow control, the virtual channel identifier, reserve

bits to be determined, a payload type, a call lost priority and a header error control field. A cell includes the header of 5 bytes and a 48 byte data content payload. It is possible to provide in one of these cells of 53 bytes, voice, video and imaging data. This cell would move at a constant bit rate speed from end station to end station and when, for example, it reached the manufacturing device the header information is discarded and the data portion of the packet is copied to the media. This media, as for example, can be CD or magnetic media for reproduction and manufacturing at this station.

Each of these cells is assigned a path by the communications network dependent upon the network traffic and load. In the communications industry this is known as fast packet switching. It provides an expeditious way to move a large volume of data.

The feature of ATM which provides for all packets or cells being of the same length presents a condition where small messages which are time dependent do not get held up by large messages which are not time dependent and can be delayed without degradation of the message's value. This is because all messages are comprised of the same 53 octet length. As hereinbefore mentioned the user has the advantage of different levels of service depending on the user's requirements for streaming the applications data. The header on each of the 53 byte uniform length cells provides routing information for each of the cells in the network. In a shared network environment, such as is present in much of today's communication networks there is uncertainty and variability. The ATM process provides for standardization in the actual transport method or transmission method through the use of fixed sized cells, plus priority routing of time critical data as specified in the cell header.

A further feature of the present invention is made evident by FIG.1, where at the master site 102, the concept of separating the different logical functions that are a part of the serving site into different or separate types of machines is presented. The idea of separating the functions allows for the scaling of different sizes of serving machines, and the specializing of the hardware configurations and even the software configuration so that they are able to perform the identified specialized functions. The machines include the network management unit 122, the directory services unit 124, the data server unit 126, the customer data base unit 128, incorporating information individualized to the customer's purchasing habits and interests and the royalty accounting unit 130. The network management unit 122 checks the status of the operation of communication links with the customer premise 110 and units within the customer premise 110. The directory for services 124 provides address functions for appropriate routing of messages requesting, for example, content data at the data server 126 or customer data in the customer database 128.

An example, illustrating the separation of function issue, is applicable to the royalty accounting unit 130 where if there is a suitable application available, for example, on an IBM AS/400 system that does credit card transactions or accounting transactions, it is desirable to adopt that kind of a machine and based on it's other characteristics, e.g., security, adapt it for use in the banking industry for use as a machine to track the royalty transactions that would flow in from the various customer premises. Similarly, a separate machine with specialized software can accomplish the network management of unit 122 at the serving site.

In addition, the data server 126 or servers can be specialized in machine type, as to the amount of storage etc. that they manage according to the type of content. So, a system can include one type of machine that is well suited to serve video data, and a second type of data serving machine that might be more suitable for such data as electronic game content data or audio data. Therefore at the different serving sites, the different functions are separated to run on separate machines, interconnecting them through switches or hubs or a suitable type of interconnecting arrangement at the serving site, and then manage the message protocols to route the appropriate message traffic coming in from the cloud 112 to the appropriate functional processing machine.

Also in FIG.1 in the description, the local site 116 and regional site 114 are not pictorially displayed like the master site 102. The bulleted list identifies some of the subset of the functions that are in the discussion breakdown of the master site 102. Certain of these functions, for example, those of customer data base unit 128, might not be of sufficiently high enough volume or traffic to justify replicating the customer data base machine 128 at all the different serving sites. So, by functionally separating the types of functions that are being served out of the sites, they can be distributed differently throughout the serving hierarchy. Therefore at some number of the serving sites or maybe only at certain levels of the serving hierarchy, some of these functions are present while at other levels they are not.

The lowest level of the data servers or ESS servers 306 in FIG. 3 are most important in terms of the high volume transactions, and replication of those is done in a more generous manner. By separating the different serving functions the opportunity is provided to choose the appropriate configuration for each machine and use the communication switch interconnections to route the data traffic to the appropriate machine or function.

FIG. 2 is an example of how the exemplary cloud 112 of FIG. 1 might physically connect three different serving sites together. In FIG. 2 there are three different service providers i.e., solid lines 202, dashed lines 204 and dotted and dashed lines 206 carriers or telephone operating companies shown by the different types of connecting lines. In FIG. 2 there are central offices (COs) 208, where the customer premises (or the stores);

are connected. Then there are trunk connections going from central offices 208 to other central offices 208 which connect to the different server sites, i.e., Master Site 210, and Regional Site 212 and Regional Site 214. Each of the server sites 210 and 212 has more than one set of trunks going to different service provider central offices (COs). In the serving sites 210, 212 or 214, the included switches are used to terminate the different trunks coming from the different central offices. On the back side of the switch is where the serving machines are connected. The switching function at the server site allows the data traffic routing between the various clouds.

In FIG. 2 there is represented three different sets of service providers 202, 204 and 206 that provide cloud function for interconnecting not only server sites, but also the retail stores.

Using ATM as the interconnect with the data server, gives the physical appearance of having the entire data content on every one of the local servers. It is still desirable to replicate some portion of that storage in each of the local servers for transaction and frequency of requestors demand reasons, but transparently, by communications through ATM, there is the appearance of having 100% of the data content of every catalog listing or item at the serving site. The number of the replication sites can be regulated or reduced depending upon how widespread the presence of ATM is.

The only reason to replicate the databank is to be able to spread the high demand and bandwidth amongst all the various sites. This practice is set forth in FIG. 2. Primary servers can function as backup servers. Such, as for example, is possible with regard to server 212 providing backup for server 214 when serving the central office 208 connected between them and illustrated in FIG. 2 as the bottom and centrally most disposed central office. The backup strategy provides for efficiency and also makes it easier to recover in the onset of breakdowns or disasters.

Being able to give the appearance of having 100% of the data content locally, camouflages having servers physically disbursed throughout a geographic area regardless of size. It is evident that through implementation of this inventive concept, directory service control function can control all the different Customer Premise sites to decide from where to access and obtain the content data. This control structure still relies on having all the data server sites connected by an ATM backbone to accomplish the necessary network.

FIG. 3 illustrates the communication network layout with emphasis on (a) a multiple-level structure of data servers providing improved reliability and availability of data, (b) redundant connections among the servers and their clients through multiple communication paths, and (c) distribution of on-line data according to demand and on-line storage and transmission costs.

In this FIG. 3 there are several repositories each of which typically contains a portion of the totality of the content material available. Each exemplary level in this structure is comprised of server class machines which are large configuration systems with large amounts of on-line DASD storage plus extensive communications capability. In the FIG. 3 example, the top most server level can contain 100% of the data which is available. The top level servers sometimes called master servers or golden masters are usually provided with 100% of the content data at very little additional cost above the 90% or other high percentage they would otherwise store as a result of allocation strategies where the content data allocated to servers at the various levels is based upon the frequency of content data use and is a function of the popularity of segments of the content data and is a result of data studies from consumer demand curves. A very high percentage also provides for redundancy and back up if service to a server should fail. Set forth for discussion purposes only, are arbitrary percentages of data divided between the multiple levels of servers. In practice the percentages are variable and are, for example, a function of such things as consumer demographics and time of year, just to mention a couple.

The servers at this level (a), are the Top Level Servers (TLS) 302. There are several of these servers, geographically placed according to several factors. The factors include: communication line availability and communication line cost, aggregate demand from the next lower level, and other considerations such as geographic and safety/security of the data.

These servers 302 have a very high percentage of the content data, sometimes 100%, but in the case of FIG. 3 it is 92% which satisfies the least demand or approximately 35% of the content data requests.

Below the Top Level Servers 302 in this structure at level (b) is a collection of servers 304 which contain a considerably lesser portion of the total data (typically around 5%). Yet, this level server 304 satisfies up to the next 20% of the demand on average, due to the popularity of the content data stored at the (b) level. These servers are the Intermediate Level Servers (ILS) 304.

Finally, in the example of FIG. 3 there is a set of servers 306 at a level (c) which support the customer premises directly. In the structure, these servers 306 satisfy the largest percent of the requests for the most frequently selected content data. These servers are the End-Site Servers (ESS) 306. As compared to the TLS and ILS, the ESS servers hold the minimal amount of the content material available, which is typically around 3%. However, as can be seen from FIG. 3, these servers 306 satisfy the bulk of the customer's requests for

content data, which is about 45%.

A preferred allocation of content data amongst a plurality of severs connected in hierachical levels includes allocating <10% of the content data at the low end data server; and allocating >80% of the content data at the data server at the high end of the hierarchy most remote from the data accessing station.

The cost analysis is driven by an effort to achieve replicating a small percentages of physical storage at the lower levels of the structure, so that a considerable amount of the transaction demand is satisfied while spreading out the balance among different server sites and machines. The demand requests for the data must be collected and an analysis made of the transaction patterns. The study results are used to minimize the data replication and maximize (i.e.,reduce) the response time of the transaction, and distribute the data for storage among large numbers of servers and sites without experiencing excessive costs to replicate these amounts of storage which are widely distributed.

For a typical request, when a data file and its content is needed and the service of a higher level server is required a request is sent from the Customer Premise (CP) 308, to the ESS 306 which is its preferred route. The communications network may determine that the requested ESS 306 is available directly (i.e., in a single path) or it may select an alternate path for this conversation. An alternate path could be selected through any path with sufficient level of service, particularly with regard to bandwidth. In this example, alternate paths could be found through any of the communication clouds 310, 312, or 314. Alternatively, when the message arrives at the selected ESS 306, the data server on that site could determine that (i) it does not have the data requested and so it could re-route the request to an ILS 304 itself or (ii) that it has too many transactions in process, and routes the request to another ESS 306 which could for example be in another geographical time zone.

FIG. 4 is an exemplary uniprocessor system data server just like a standard personal computer IBM PS/2 Model 95 with a single processor 402 connected through a fast bus 406 to some amount of memory 404, a redundant array of independent disks (RAID) 416, SCSI buss 408 to hang storage units 412 on and connecting line 410, then some sort of I/O 414 which represents things like communications adapters.

FIG. 5 illustrates a multi-processor data server system where there is more than one processor 502 sitting on a single fast bus 504 or interprocessor memory bus of some sort with multiple I/O buses 506, 508 and 510 hanging off of the system. There are several different types of machines that represent this kind of structure ranging from the IBM 3090 types of mainframes to systems like Tricord, Sequent and Pyramid Models. There are many kinds of multiprocessor shared memory types of machines.

FIGS. 6 and 7 illustrate multiple processing elements or servers 602, such as those illustrated in FIGS. 4 and 5 that are interconnected with different possible high-speed interconnects. The first example of interconnects, as set forth in FIG. 6, is LAN technology interconnecting various processing elements with their own private memories and I/O devices or servers 602. This is also illustrative of a fiber channel standard or a proprietary interconnect schemes as shown by way of the oval interconnect 604.

FIG. 7 is another distribution topology showing that using an ATM or other high speed switch 702 as an interconnect method between those different types of processing elements or servers 602. In shared memory multiprocessors, blocking effects and interference, from trying to access the same memory, are experienced. Bottlenecks on the bus or in the memory are the more adverse conditions which can arise and are adverse to the speed of the processors and the I/O devices. However, shared memory does simplify some of the software structures. Since all the data is accessible from the various and different processing elements in the distributed function structures, there is less interference, because there is more message-based systems which allows them to run asynchronously. But under such circumstances, latency or the time interval in terms of requesting the data for transfer and getting it transferred across whatever the interconnect scheme is, is of prime concern. The physical topology is important but it usually results in the need for software which is more complex. There is a need to acquire the right message constructs with low latency, so that the form of throughput is acceptable. The other benefit is that structures of the type in FIGS. 6 and 7 tend to scale up from small configurations to the large and super large configurations with less difficulty than the shared memory configurations. In shared memory sooner or later bus limitations or locking interferences are presented which are difficult to overcome. So the preference is a methodology that permits scale-up without undue complexity or waste.

Another flexibility with such structures is the ability to specialize the different types of processors, so that a special assist for doing communications and a special assist for doing the data storage retrieval can be created. Also specializing the different processing elements can be done, as opposed to using a single homogenous type of processing element which is what is typically done in the shared memory multiprocessor. For the loosely coupled structure of FIG. 7, each of these elements can also be a tightly coupled shared memory processing element itself.

FIGS. 8, 9 and 10 relate to switching, routing and other adapter and computer techniques with respect to the service providers as herein before presented regarding FIGS. 1 and 2. These switching techniques provide

the means for gathering the incoming and outgoing messages. The incoming messages are included within a trunk, which is a high speed aggregated collection of bandwidth. The speed is higher than the consuming side of the system and its total bandwidth is for all the end terminals that are in the store or customer premise. The outgoing messages are essentially the reverse process and which exit into the network in a cloud such as cloud 112 of FIG. 1.

The trunk in the case of FIG. 8 ranges in speed from a T1 to a T3, or an ATM and operates over fiber type of physical connectivity. If the switch 802 is the connection media, then messages are switched from the trunk to any one of several other communication ports. The advantage of a switch is that typically the switch 802 is designed for a large aggregate amount of bandwidth which allows a large number of simultaneous connections to occur, including on up to the bandwidth limit. A less expensive alternative to FIG. 8 is the configuration of FIG. 10 which shows a computer 1002 acting as a switching mechanism or a gateway where the trunk is coming in through a trunk adapter 1004 plugged in on the computer bus. In this example, it could be either a T1 adapter, a T3 adapter or an OC-3 fiber adapter plugged in on the computer bus with other adapter cards plugged into the computer bus, that, for example, could be LAN types of adapters. The computer 1002, with its software and hardware perform a multiplexing, a routing, or a gateway type function from the LAN conversations to the trunk adapter.

FIG. 9, is an example of specialized hardware built to represent the same kind of functionality. There are several manufacturers providing products identified as either hubs or routers that perform similar functions, where there is one type of physical connectivity that is routed or bridged to LANs.

All three of these figures, FIGS. 8, 9 and 10, represent the exemplary arrangement of an incoming high speed trunk 118 in FIG. 1 into the customer premise. The switching function occurs whether at an actual switch or hub router or even a computer acting as a router, to take the traffic coming in from the outside network and then switch it to the appropriate data consumer and or terminals, inside of the customer premise.

Typical switches are made out of what is referred to as switching fabric and typically are of a cross point switching mechanism. The actual physical medium for the higher speed, transmission may, be on copper or on fiber. Although there are other types of fabric that are used to perform switching functions, they usually provide some sort of a simultaneous connection between an incoming port and an outgoing port. Crosspoint switches are a space division form of multiplexing technique. Either the hub router or the computer gateway/router technique, are typically time division type of multiplexing mechanisms.

In the case of FIG. 10, the computer could be a personal computer, a workstation, or almost any computer that has the normal input/output bus concept with I/O adapters plugged into it. In such a case the time division multiplexing aspect of the computer bus to switch data from one adapter to another is employed. The hub and the router are specially packaged versions of the computer of FIG. 10. Alternatively, they can include a switching fabric, i.e., a space division type of fabric. There are three types of multiplexing, i.e., frequency division, space division or time division.

One of the distinctions of FIG. 11 is the cloud 1102 between the regional server site 1106 and the retail store site 1108. The service provider can be any from the group including a local phone company, operating company, an interexchange carrier, or a bypass company. Examples of the communication media in clouds 1102 and 1104 are copper wire, under sea cables, fiber, microwaves and satellite links.

Typically the T carriers or the DS, digital signal hierarchies are carried over microwaves, as well as satellites. Presently some satellite companies are exploring how to transport the higher speed carriers like the OC-3 types of speeds across their carriers so that ATM can flow at higher speeds than 44 megabits per second.

FIG. 12 details the manufacturing and decompression aspects of the real-time transfer of data to the manufacturing subsystem 104 from the repository. Here the aggregate data coming in from the cloud 1102 could be ATM at 155 megabits per second, but the data going out to each one of the manufacturing machines 1202, 1204, 1206 and 1208 depends on their individual requirements. In this specific unit 118, i.e., an ATM switch, it could in fact, be the PC that is used to de-multiplex the data and route it to each one of the manufacturing machines at their different rates. Data comes in at real-time and is transmitted at whatever the requirement is for each of the machines 1202, 1204, 1206 and 1208, respectively. This is accomplished whether it is at 1.4 megabytes per second or 1.2 megabytes or if it was 700KB per second. The printer 1206 has a lesser need for real-time data flow and the data transmission can be at a specific data signalling rate.

In the preferred embodiment on-demand content data system there are a multitude of different data rate values for the various different medias, techniques, compression algorithms of units 1210, 1212, 1214 and 1216 and reproduction machines. They are variable, and dependent on the streaming requirements of the reproduction machine, a function of whether it is audio, or video and audio, or if it is video compression by compression technique.

The preview station has a different required data flow rate then the CD writer. A CD writer, for example, can write six times normal playback speed. A high speed data rate of 1.1 megabytes per second is necessary

for a six times CD writer, 1.4 megabytes for an eight times tape writer and 7 megabytes for 40 preview booths playing 1.4 Mbit per second MPEG compressed video or CD quality audio.

It is desirable to encrypt certain content data to maintain a higher level of security for proprietary information and communication channels and communication switching techniques which are readily accessible to unauthorized or unintended interception. According data may be encrypted for example at Data Capture unite 132. Data is encrypted at unit 132 with a random pattern of characters known as an encryption key. Each cryptographic key is protected and is stored in royalty accounting unit 130. Decryption of the encoded data is accomplished at the units 1210, 1212 and 1214 where the decryption and decompression steps are performed in a manner consistent with the processes of encryption and compression at the time of data capture. Units 1210, 1212 and 1214 also provide the capability of buffering to accomplish streaming to the CD writer 1202, tapewriter 1204, color printer 1206 and alternate unit 1208.

A preview station 1303 of FIG. 12 has a need for video considerations. In some cases it will have MPEG video compressed and alternatively RTV or PLV compressed DVI or any of the various compression formats supported by software only motion video playback. For effective streaming, the conditions must exist for continuous transmission once commenced, until completion of the data segment, which is usually the end of the recording or data stream. This continuous transmission must be guaranteed not to fall below the speed required by the playback device to sustain continuous full motion video.

The communications asynchronous transfer mode (ATM) with its switching technique provides advantages over broadband communications, and it's incorporation within the present invention presents opportunities for handling all types of traffic and is protocol insensitive because of the bandwidth and the high speed bit rate. It is particularly suitable for voice, video, image and data transmission as identified in the present invention.

Although the operational aspects of the innovative on-demand digital information system has been discussed in part with regard to Figs. 1-12 and the corresponding communication and network features of the present invention, Fig. 13 and its description presents a total overview of a typical preview, order, authorization, manufacture and purchase process for the average customer. The total process has been divided into component substeps identified as the Order Process (OP), the Authorization Process (AP), the Manufacturing Process (MP) and the Purchase Process (PP). The integral and alternate steps of each of the OP, AP, MP and PP processes are identified by general steps which are sequentially numbered with the lead letters a, b, c and d, respectively. These sequenced steps identify the logic flow of a typical order from the initial browsing through to the purchase.

Certain terms, as defined below, will aid in the understanding of the logic and flow of the order and its subsequent handling as hereafter described for FIG. 13. The definitions include Transactional Terms, Database Tables and Process Messages.

Transactional Terms:

Accounting Server - a set of computer programs which can and typically do serve as the central point of control within a store. These programs provide the central data base for all orders originating in a store, perform all financial accounting (related to orders placed through this system) for a store, and perform all authorization interchanges with the Authorization Server.

Authorization Server - a set of computer programs which makes decisions, authorizing or denying approval to manufacture a product.
These programs maintain several databases such as:

- a database of products available;
- a database of royalties owed to the company, such as a label company, owning a product; and
- a database providing criteria for determining if a given retail outlet is authorized to sell a given product.

These programs provide a decision making process for several stores. These programs typically execute at a site that is separate from a retail store.

Customer Premise - that location from which a customer can interact with the on-demand system, including, but not limited to, such interactions, like previewing, ordering, product pick-up and maintaining transactional records.

Device Controller - a set of computer programs which control the manufacturing process. This server is located in a retail store.

Item - a manufactured product or work, suitable for purchase. As an example, it is that which a customer might procure in a traditional purchase. It includes primarily the media on which the information is recorded plus secondary materials such as a case, labels, pamphlets, case liner, etc. In an exemplary retail system, an item

would be a compact disc, a cassette tape or electronic video games contained in a package along with appropriate labels and printed materials.

Line Item - a record description of a single work, in some quantity, which is to be purchased. A Line Item consists of one or more identical Items. In addition to a count of the number of copies desired, is an indication if compact disc or cassette tape is to be used as the recording media.

Order - a collection of Items a customer wishes to purchase. Orders are made up of one or more Line Items.

Piece - one component part of an Item, such as:

- the pamphlet placed inside a compact disc case,
- the label placed on the case, or
- the tape cassette.

Point of Sale - a terminal such as a cash register, located at the point where a customer makes a purchase. This terminal includes a computer program that allows a sales person to view, and, if appropriate, change an order.

Preview Station - a location, including devices with which an individual accesses a sample of a work. Typically the access is to demonstrate the capabilities of the offered product at a preview by the viewer with the prospect of leading to a sale of the product to that individual. A station of this type would generally be located in a retail sales environment such as a store or kiosk. Orders for product may be placed at the Preview Station.

Work - an Item being offered for preview and/or for sale.

Database Tables:

Grant Table - a persistent record of all manufacturing transactions requested on a per store basis. Contains one record for each Item ordered describing the authorization status of that Item.

Line Item Table - a persistent record of all ordered Line Items for all orders on a per store basis.

Make Table - a temporary record of all Items ordered but not yet manufactured. There is one record for each Item describing what is to be manufactured and the stage of the manufacturing process.

Orders Table - a persistent record of all Orders on a per store basis.

Process Messages:

op_approved - the message sent from the Authorization Server to the Accounting Server when an Item is approved for manufacturing.

op_cancel - the message sent from the Point of Sale station to the Accounting Server to indicate that an order has been canceled prior to the authorization request and the order should be removed from the system.

op_cannot_make - the message sent from the Device Controller to the Accounting Server when a Piece of an Item has been unsuccessfully manufactured.

op_denied - the message sent from the Authorization Server to the Accounting Server when an Item is not approved for manufacturing.

op_grant - the message sent from the Accounting Server to the Authorization Server requesting authorization to manufacture a single Item.

op_grant_failed - the message sent from the Accounting Server to the Point of Sale station when an authorization request is denied by the Authorization Server.

op_made - the message sent from the Device Controller to the Accounting Server when a Piece of an Item has been successfully manufactured.

op_make - the message sent from the Accounting Server to the Device Controller to indicate what Items are to be manufactured.

op_make_failed - the message sent from the Accounting Server to the Point-of-Sale station when an Item is unsuccessfully manufactured.

op_ok - the message sent from the Point-of-Sale station to the Accounting Server to indicate that an order has been validated, and paid for, by the customer.

op_order - the message sent from the Preview Station to the Accounting Server to indicate that an order has been entered into the system.

op_order_complete - the message sent from the Accounting Server to the Point-of-Sale when all Items in an order have been completed either successfully or unsuccessfully.

Order Process

a01) The customer, as a prospective purchaser, enters the store, and accesses one of a multitude of multimedia preview stations 1303, at a preview subsystem 1302 as shown in FIG. 13. In this example, the preview station is the Customer Premise. The customer inserts a membership card into a reader as more fully explained for FIGS.16, 17 and 28 and sets about previewing various works. Through the touch screen panels provided as shown in part and by way of example in FIGS. 18-27, the customer can choose to preview and listen to certain works, or to listen to works while viewing an accompanying video. A screen icon of a pair of headphones 2402, as seen in FIG. 24 and another icon of a TV set 2404 are used to inform the customer of the availability of preview materials. When the customer has made a purchase decision, commands at touch pads 2406 or 2408, describing the format of the desired items, are entered on the touch screen.

a02) A Preview Station 1303 of FIG. 13 prints a memorandum of the Order 1602 for the customer, as shown in FIG. 16.

a03) The Preview Station of the subsystem 1302 forwards a message describing this order (i.e., message type op_order) to the Accounting Server 1304 of the accounting and management subsystem 1306.

a04) The Order is stored in the Orders Table database on the Accounting Server 1304. The order information includes:

- preview station number
- date and time of day
- a Line Item Table describing the details of the purchase, including:
  - artist
  - title
  - number of copies
  - media: compact disc or cassette tape.

a05) The customer approaches the checkout area of the store included within the accounting and management subsystem 1306, and hands a personal membership card or other suitable identifying card to the sales clerk.

a06) The sales clerk inserts the customer's membership card through a card reader at the Point-of-Sale (POS) terminal 1308.

a07) The POS 1308 sends a remote database query message to the Accounting Server 1304 providing this customer number.

a08) The Accounting Server 1304 searches the Orders Table database for this customer's Order. If the Order cannot be located, the Accounting Server 1304 returns an error to the POS 1308. However, if the Order is located, the Accounting Server 1304 returns a copy to the POS 1308 for display. Included is an Order number identifier.

a09) Once the customer's Order has been displayed at the POS 1308, the sales person will ask if the customer wishes to make any changes. Changes which are possible include:

- change quantity of any Line Item;
- delete Items;
- change format of ordered Items (e.g., compact disc to cassette tape or cassette tape to compact disc);
- add new Items not previously entered;
- cancel the entire Order.

a10) As an alternative to the above POS purchasing sequence, the customer could simply approach a sales person immediately upon entering the store and have the sales person enter an Order. In this case, the procedures related to the activities at the preview station subsystem are not necessary. (The Order will not be found at step a08 but the sales person will be able to enter the order in its entirety at step a09.)

a11) The customer now makes payment via cash, check, or credit card, according to the accepted procedure for each particular store or retail outlet.

a12) When the sales person completes collection of the payment, the POS 1308 will send an op_ok message to the Accounting Server 1304. At this point, all the information about the Order is in the on-demand digital information system. The system is now ready and will automatically proceed to authorize the Order, to credit royalties to any party or entity so entitled as set forth in pre-existing agreements, and to issue manufacturing orders to generate the customer's selections.

Authorization Process

b01) When the Accounting Server 1304 receives the op_ok message, it replaces the copy of the Order in the Orders Table database. This is to account for any changes made by the sales clerk at the POS terminal 1308.

b02) The Accounting Server 1304 updates the Grant Table and Make Table with one record per Item, initialized to blank.

b03) Before any manufacturing can begin, an authorization must be obtained and an Authorization procedure is performed. This insures that only works authorized or approved for manufacture are manufactured and that any and all royalty requirements are correctly recorded.

b04) The Accounting Server 1304 sends an op_grant message to the Authorization Server 1312 of the Authorization Server subsystem 1310 for each Item in each Line Item. (Note: If this step is reached due to a manufacturing failure c13 step, only the failing Items will have grant requests sent.)

b05) The Authorization Server 1312 performs various authorization checks on each op_grant message and returns an op_approved or op_denied message to indicate ok/not ok to manufacture that particular Item.

b06) The criteria for denying authorization may be objective or may include subjective rules.

Exemplary criteria include:

- release date is later than today's date;
- a particular store's credit rating is exceeded;
- certain retail outlets may want to prohibit sale of certain works;
- a content provider's or label company's unique rules; and
- copyright license for a work does not exist in the country where the store is located.

b07) If an item is rejected for any reason, an op_denied message is returned for that Item by the Authorization Server 1312 to the Accounting Server 1304.

b08) When op_denied is received for an Item, the Accounting Server 1304:

- updates the Grant Table, Line Item Table, and Order Table reflecting what was denied; and
- the denied item is removed from the Make Table.

b09) Since more than one Order at a time can have outstanding authorization requests, if this is the last Item in a particular Order, the Accounting Server 1304 marks the Order complete and sends an op_make message to the Device Controller 1314 in the Manufacturing Subsystem 1316.

b10) If there are more items in this Order, the Accounting Server 1304 loops back to Authorization Process step b07).

b11) If an op_approved message is received for a Line Item, the Accounting Server 1304 updates the Grant Table, Line Item Table, Make Table, and Orders Table reflecting what was approved. It will then loop back to Authorization Process step b09.

Manufacturing Steps

In the specification below, the Device Controller 1314 configuration depicted in Fig. 13 is representative of those machines which are necessary to operate the manufacturing machines, such as the CD writer 1318 and the tape writer 1320 in a particular retail store. The actual number of manufacturing devices and their location may differ from that illustrated in FIG. 13 based upon a given store's requirements.

c01) When the Device Controller 1314 receives the op_make message from the Accounting Server 1304 the Order contains only those Items which have been authorized and are to be manufactured.

c02) The Device Controller 1314 displays the Order as a set of manufacturing instructions on the Manufacturing Technician's Terminal (MTT) 1322. Each Item to be manufactured is displayed as a single line.

c03) In the case of media manufacturing, when the appropriate machine such as the CD writer 1318 or tape writer 1320 is ready, the technician presses a FETCH icon on the MTT 1322. This sends a message to the respective Device Controller 1314 for that manufacturing machine. The technician then presses a START icon on the MTT 1322 to initiate actual manufacture.

c04) The Device Controller 1314 will then establish a dialogue with the Data Server 1324 of the Data Server Subsystem 1326 and begin the content data transfer to the exemplary machine 1318 or 1320 indicated in the Item.

c05) For any printed material required, the MTT 1322 sends a message to the Device Controller 1314 identifying the data for printing..

c06) The Device Controller 1314 will then establish a dialogue with the Data Server 1324 and begin transferring the printable material data to the printer 1328.

c07) In both manufacturing steps c04 and c06, the data being transferred may be encrypted or compressed or both or neither. Before the data is actually routed to the manufacturing machine such as machine 1318, 1320 or 1328, a decryption and/or decompression step is likely necessary. The cryptographic key needed for the decryption step may be contained in the original op_approve message of Authorization Process step b11.

c08) For each Piece of an Item successfully completed, the Device Controller 1314 sends an op_made message to the Accounting Server 1304.

c09) The Accounting Server 1304 notes each op_made message:

- when all pieces in an Item are made successfully, that item is marked complete;
- when all Items in a Line Item are complete, that Line item is marked complete; and
- when all Line Items in an Order are complete, that Order is marked complete.

c10) If an error occurs at any step of the Manufacturing Process, the recovery steps depend on the failure point:

- for a failure in the printing step, the technician must manually retry that step.

c11) For a failure in writing a media device such as at units 1318 or 1320 that Item will be scrapped and a new authorization sequence must be initiated.

c12) If any Piece of an Item does not complete successfully, the Device Controller 1314 sends an op_cannot_make message to the Accounting Server 1304.

c13) The Accounting Server 1304 receives the op_cannot_make message. The order number is used to retrieve the original Order from the database. If the failing Piece is media, as opposed to printed matter, a new grant request is constructed for the failing Item and the processing resumes above at step b04 of the Authorization Process. This step and the associated authorization grant / royalty accounting / manufacturing steps above step b04 through step c13 are repeated 'n' times - where n is a retry count that is stored in an operating characteristics file in the Accounting Server 1304. When a failure has been retried 'n' times, the Accounting Server 1304 will make a final determination that this Item cannot be delivered and processing will continue at Purchase Process step d01.

c14) The original Order remains on the account for the requesting retail store, i.e., normal royalties remain charged against that store. That store manager must subsequently process a Return Manufacturing Authorization request to obtain credit for the failed Pieces. This insures that no duplicate media may be produced without royalty payments being accrued to the content owner.

Purchase Process

d01) Once the Accounting Server 1304 has received a response for all Pieces of all Items in the Order (either op_made or op_cannot_make and the retry count is exhausted), it accesses the original Order and processes the Items from top-to-bottom matching as many Line Items requested to those completed.

d02) Any Items whose retry counts were exhausted [see Manufacturing step c13)] must have their selling amounts backed out from the total charges for the Order. A credit will need to be issued, for a charge sale, or a cash refund, for a cash sale, according to the respective store's operating policy.

d03) The Accounting Server 1304 makes a final update to the Order so that it reflects what is actually being delivered to the customer. A message is sent to the POS 1308 that the Order is complete.

d04) When the POS 1308 receives this message, it queries the Accounting Server 1304 for a copy of the updated Order.

d05) The Accounting Server 1304 sends a copy of the updated Order to the POS 1308.

d06) The POS 1308 prints the final receipt and the sales person is informed (by a message to the screen) and notifies the customer that their Order is ready for pickup.

FIG. 14 is a functional block diagram of the present inventive on-demand digital data system and is distinguishable from the foregoing figures as hereinafter mentioned. It identifies certain products or manufacturers as exemplary sources of devices for practicing the present invention. Manufacturing controller unit 1402 controls the two CD writers 1318 described in FIG. 13 and identified as CD writer unit 1202 in FIG. 12 in a manner to operate the two machines at two different speeds, i.e., twice real-time (2x) and 6 times real-time (6x). Suitable CD writers are available from various manufacturers including the Eastman Kodak company. Manufacturing controller unit 1404 controls the A/D interface 1410 for the special purpose devices built for what is tape writer 1320 of FIG. 13, as would a controller 1314 for the tape writer interface 1330 in FIG. 13. Manufacturing controller unit 1404 is also the interface for the Sun Microsystem's Workstation unit 1412 and a specific Xerox color copier or color printer 1328 of FIG. 13. FIG. 14 illustrates the Manufacturing Control Console 1406 optionally connected to the Retail Host Server 1408.

FIG. 15 has been discussed in some considerable detail with relation to FIG. 13. Further, it shows, by way

of example only, certain illustrative programs or systems such as PARADOX which is generally hereinbefore referred to as a database. The Presentation Manager program shows that on-demand systems can be made to operate with the OS/2 operating system, as one such system. Likewise the booth 1502 is a multimedia preview station.

The communication process and the flow of messages or conversation between the subsystems of a typical on-demand system was described with regard to FIG. 13. In FIG. 15 there is provided a block diagram of a typical system encompassed by that of FIG. 13, and illustrates the presence of the communication engines of each of the typical subsystems, i.e., the Preview Booth station 1502, the Retail Accounting station 1504, the Central Host Accounting station 1506, the Manufacturing Control station 1508 and Point-of-Sale station 1510.

For the purpose of this presentation, a conversation is as a group of commands and responses that are logically related. For example, the set of commands and responses exchanged by the Retail Accounting subsystem 1504 and the Manufacturing Control Application 1508 is know as the 'Manufacturing Conversation'. The five clearly defined conversations of FIG. 15 consist of bi-directional, variable-length, point-to-point messages. These messages can be transported by any suitable communications protocol such as NetBios, TCP/IP, SPX/PIX, etc. Guaranteed delivery is a requirement of the communications protocol selected.

Consider the following conversations:

Conversation	Description
Grant	Established between the Retail Accounting 1504 and the Central Host Accounting 1506 for the purpose of requesting and granting the manufacture of a music or audio product.
Manufacturing	Established between the Retail Accounting System 1504 and the Manufacturing Control Application 1508 to submit an order to reproduce a particular audio or video product.
Stream	Established between the customer preview station (PS) and the file server in order to transfer audio and video data to the PS for user interaction. This same conversation is also established between each device controller machine and the Data Server to transfer data to the CD and tape writers as illustrated by way of example, in more detail in FIGS. 11, 12, 13 and 14.
Order	This conversation is established between the PSs and the Retail Accounting System 1504 for the purpose of placing and order put together by a customer.
Device	This conversation is established between the Manufacturing Control Application 1508 and a device server. Typical device controllers are CD writers, tape writers, and color printers, as shown in FIG. 12. This conversation is strictly used to transfer control command to and from the device controllers. The device controllers themselves establish a Stream conversation of their own with the Data Server.

In order to provide for platform and operating system independency, the unsigned long type in all of our data structures is preferred. The unsigned long data type can be easily manipulated by DOS, OS/2, Windows, and UNIX or X Windows, or other operating environment.

The message structure is such that each message consists of a header followed by message-dependent information. The handler of a particular message knows about the contents of the message-dependent section of the message. Message handlers are hereinafter discussed. The message is structured as follows:

```
typedef struct _msg
      {
      unsigned long dwOpCode;
      unsigned long dwSequence;
      unsigned long dwFlags;
      unsigned long dwBuffLen;
      unsigned char Buffer[1];

      } MSG, *PMSG;
```

A common communications application program interface (API) used throughout an exemplary on-demand system assumes that a Netbios transport layer is available, or that a Netbios emulation can be readily implemented. Such is the case for NetWare, Banyan Vines, and other network operating systems.

Further, the communications engine hides the sequencing of the messages as well as retries and re-connections.

Incoming messages have the following data structures:

```
/*
 * The following typedef defines a pointer to a function that handles
incoming messages
 */

typedef unsigned long (*PMHFN)(unsigned long dwSessionId,PMSG pMsg);

typedef struct _msg
      {
      unsigned long dwOpCode;
      unsigned long dwSequence;
      unsigned long dwFlags;
      unsigned long dwBuffLen;
```

```
        unsigned char Buffer[1];

        } MSG, *PMSG;

typedef unsigned long LISTHANDLE;

// Error codes

#define ERROR_ALREADY_INITIALIZED       1001
#define ERROR_MEMORY_ALLOC              1002
#define ERROR_NOT_INITIALIZED           1003
#define ERROR_INVALID_PTR               1004
#define ERROR_HANDLER_NOT_FOUND         1005
#define ERROR_LANRESOURCES              1006
#define ERROR_NAME_NOT_REGISTERED       1007
#define ERROR_NO_MORE_NCBS              1008
#define ERROR_SUBMITING_NCB             1009
#define ERROR_CONNECT_IN_PROCESS        1010
#define ERROR_NOT_CONNECTED             1011
#define ERROR_CONNECTING                1012
#define ERROR_SENDING                   1013
#define ERROR_INVALID_SESSIONID         1014
#define ERROR_COMM_RESOURCES            1015
#define ERROR_SEND_TIMEOUT              1016
#define ERROR_TOO_MANY_COMMANDS         1017

#ifdef OS2_COMMSYS
#define ERROR_ACCESSING_SEM             2001
#endif

// Session flags

#define SF_CONNECTED                    0x00000001
#define SF_CONNECTING                   0x00000002
#define SF_PENDRECEIVE                  0x00000004
#define SF_PENDLISTEN                   0x00000008
#define SF_PENDCALL                     0x00000010
```

The following is a continuation of the description for the programming interface. More detail about the various conversations are hereinafter discussed. For example, later sections describe the conversation-dependent part of the message as it is defined for each conversation.

Initialization of the communications engine is necessary by each process that uses its services.

The global default message handler and the global exception handler MUST be passed to this function.

NULL values are not allowed. If this function fails the Comm Engine cannot be used. It is necessary to check the return code.

```
int InitCommSystem(           // Initialize the system, create chain
      int adapters,           // Number of adapters in this machine
      PMHFN pfnDefMsgHandler, // Pointer to global def msg handler
      PMHFN pfnExceptionHandler);// Pointer to global exception handler
```

Each message is handled by a message handler, that is, when a message is received the proper message handler is invoked. If there is no message handler defined for the incoming message the Communications Engine invokes the default message handler for that conversation (or session) if one has been defined. If one has not been defined the Comm Engine invokes the global default message handler which is guaranteed to exist.

The following functions hide the implementation details of message handler support and allow the creation and maintenance of handler lists. The first step in creating a session is to create a handler list and adding the various message handlers to it.

```
int CreateMessageHandlerList(       // Create an arbitrary handler list
      LISTHANDLE *hList);           // Return id of arbitrary list here

int InstallMessageHandler(    // Add a message handler to the list
      LISTHANDLE hList,       // ID of arbitrary handler list
      unsigned long dwOpCode, // Opcode of handler to be installed

PMHFN pfnMessageHandler);     // Pointer to function handling message

int ReplaceMessageHandler(    // Replace an existing mesage handler


LISTHANDLE hList,             // ID of arbitrary handler list
unsigned long dwOpCode,       // Opcode of handler to be replaced
PMHFN pfnMessageHandler);     // Pointer to function handling message

int RegisterDefaultMessageHandler(  //Install default handler for this list
      LISTHANDLE hList,             // ID of arbitrary handler list
      PMHFN pfnDefaultHandler);     // Pointer to default handling functn

int RegisterExceptionHandler(       // Install default exception handler
      LISTHANDLE hList,             // ID of arbitrary handler list
      PMHFN pfnExceptionHandler);   // Pointer to exception handling
function
```

The following functions form the core of the Comm Engine. These functions are used to create server ses-

sions as well as to connect with remote server sessions.

```
int RegisterLocalName(              // Register a name a-la-netbios
      short iAdapter,               // Network adapter number (0 through 4)
      unsigned char *pszLocalName); // Name to be registered

int CreateServerSession(                  // Initiate and receive all
      unsigned char *pszLocalName,        // Name client connects to
      unsigned char *pszRemoteNames,   // Name expected to connect ("*" for
ALL1)
      LISTHANDLE hList,             // ID of  handler list
      unsigned long *pdwSessionId); // Pointer where session id returned

int ClientConnect(                  // Client connects to server
      unsigned char *pszLocalName,  // Name of this client
      unsigned char *pszRemoteName, // Name of the server session
      LISTHANDLE hList,             // ID of handler list
      unsigned long *pdwSessionId); // Pointer where session id returned

int TerminateSession(               // Ussually client terminates session
      unsigned long dwSessionId);   // Id of session to be terminated

int QuerySessionStatus(             // Query the status of a given session
      unsigned long dwSessionId);   // Id of session to be queried
```

The transport manager section includes three functions. These functions allow a message to be received and its message handler invoked, as well as sending a message across the link.

```
int SendNetMessage(                    // Client / Server sends message
      unsigned long dwSessionId,       // Id of established session
      unsigned long dwOpCode,          // Opcode to send to receiver
      unsigned long dwFlags,           // Value of flags to send to receiver
      unsigned long dwBuffLen,         // Length of buffer to send
      void *pBuffer);                  // Pointer to buffer data to send

unsigned long QuerySessionStatus(      // Returns session status flags
      unsigned long dwSessionId,       // Id of established session

 int DispatchNetMsg(                   // Invokes message handlers
      int bMainLoop);                  // Must be TRUE if called from main
loop, FALSE otherwise
```

The previous function deserves a special explanation. This function allows to "cooperatively" dispatch messages to the appropriate handlers and the TRUE value is only used when DispatchNetMsg is called from within the main loop.

For example, the following code should replace your traditional GetMsg/DispatchMsg loop in an OS/2 Presentation Manager application:

```
bDone = FALSE;

while (!bDone)
      {
      fResult = WinPeekMsg(hab,(PQMSG)&qmsg,(HWND)NULL,
            (ULONG)NULL,(ULONG)NULL,PM_REMOVE);

      if (fResult)
            {
            if (qmsg.msg == WM_QUIT)
                  bDone = TRUE;
            else
                  WinDispatchMsg(hab,(PQMSG)&qmsg);
            }


      DispatchNetMsg(TRUE);
      }
```

A similar approach should be used under MS Windows.

If you are using the Comm Engine under PC DOS or OS/2 (non-PM application) you should have a loop in your main as follows:

```
while (TRUE)
      {
      DispatchNetMsg(TRUE);
      }
```

If processing of a particular message will take a long time (more than one second) you should call DispatchNetMsg from within the handler. Make sure you call it with its parameter set to FALSE as follows:

```
MessageHandler1(...)
{
      for (i = 0; i < n; i++)
            {
            // the following process takes a little while to execute
                  .
                  .
                  .
            DispatchNetMsg(FALSE);
            }
}
```

The following example uses a hypothetical sample conversation with two (2) opcodes between a server application and a client application. This code ignores run-time errors for the most part.

For example if it is desired to start a server and a client in machine #1, and another server and another client in machine#2 type the following:

At the OS/2 prompt in machine #1:

```
sessions -Ss1.c3 -Ss2.c4 -Cc1.s3 -Cc2.s4
```

At the OS/2 prompt in machine #2:

```
sessions -Ss3.c1 -Ss4.c2 -Cc3.s1 -Cc4.s2
```

The command-line option -S defines a server and -C defines a client. For example, -Ss1.c3 defines a server named s1 that accepts a connection from a remote client named c3. Option -Cc4.s2 defines a client named c4 that connects with a remote server named s2.

```
#pragma pack(1)

#define INCL_DOSPROCESS
#include <os2.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <malloc.h>
 #include <memory.h>
#include <ctype.h>
#include <time.h>

#include <commsys.h>

int main(int argc,char **argv);
int DefMsgHandler(unsigned long dwSessionId,PMSG pMsg);
int ExceptionHandler(unsigned long dwSessionId,PMSG pMsg);
int MessageHandler1(unsigned long dwSessionId,PMSG pMsg);
int ClientHandler1(unsigned long dwSessionId,PMSG pMsg);


#define     OP_REQUEST1       100
#define     OP_SERVICE1       200

int main(int argc,char **argv)
{
      int rc = 0, i, j;
      LISTHANDLE hList = 0;
      LISTHANDLE hCList = 0;
      char *p;
```

```
        unsigned long dwServerSessions[10];
        unsigned long dwClientSessions[10];
        int nSSessions = 0;
        char szClientName[32];
        int nCSessions = 0;
        PMSG pMsg;
        char szArg[32];

        if (argc < 3)
              {
              printf("usage: SESSIONS -cClient1.Server1 -cClient2.Server2 -
sServer1.Client?... \n");
              DosBeep(1000,100);
              return(-1);
              }

        rc = InitCommSystem(1,DefMsgHandler,ExceptionHandler);
        if (rc != SUCCESS)
              printf("Initializing Comm System (%d)\n",rc);
        if (rc != SUCCESS)
              return(DosBeep(1000,100));

        for (i = 1; i < argc; i++)
              {
              strcpy(szArg,argv[i]);
              if (szArg[0] == '-')
                    {
                    if (toupper(szArg[1]) == 'C')
                          {
                          p = (char *) strchr(szArg,'.');
                          if (p == 0)
                                {
                                printf("Invalid command line argument\n\n");
                                return(DosBeep(1000,100));
                                }
                          *p = '\0';
                          strcpy(szClientName,&szArg[2]);
                          *p = '.';
                          rc = RegisterLocalName(0,szClientName);
                          if (rc != SUCCESS)
```

```
                                printf("Registering %s
(%d)\n",szClientName,rc);
                        if (rc != SUCCESS)
                                return(DosBeep(1000,100));
                        }
                else if (toupper(szArg[1]) == 'S')
                        {
                        p = (char *) strchr(szArg,'.');
                        if (p == 0)
                                {
                                printf("Invalid command line argument\n\n");
                                return(DosBeep(1000,100));
                                }
                        *p = '\0';
                        strcpy(szClientName,&szArg[2]);
                        *p = '.';
                        rc = RegisterLocalName(0,szClientName);
                        if (rc != SUCCESS)
                                printf("Registering %s (%d)\n",&szArg[2],rc);
                        if (rc != SUCCESS)
                                return(DosBeep(1000,100));
                        }
                }
        }

    rc = CreateMessageHandlerList(&hList);
    if (rc != SUCCESS)
        printf("Creating message handler list (%d)\n",rc);
    if (rc != SUCCESS)
        return(DosBeep(1000,100));

    rc = InstallMessageHandler(hList,OP_REQUEST1,MessageHandler1);
    if (rc != SUCCESS)
        printf("Installing message handler 1 (%d)\n",rc);
    if (rc != SUCCESS)
        return(DosBeep(1000,100));

    rc = RegisterDefaultMessageHandler(hList,0);
    if (rc != SUCCESS)
        printf("Registering default message handler 1 (%d)\n",rc);
```

```
        if (rc != SUCCESS)
              return(DosBeep(1000,100));

        rc = RegisterExceptionHandler(hList,0);
        if (rc != SUCCESS)
              printf("Registering default exception handler 1 (%d)\n",rc);
        if (rc != SUCCESS)
              return(DosBeep(1000,100));

        // Create local server sessions

        for (i = 1; i < argc; i++)
              {
              strcpy(szArg,argv[i]);
              if (szArg[0] == '-')
                    {
                    if (toupper(szArg[1]) == 'S')
                          {
                          p = (char *) strchr(szArg,'.');
                          if (p == 0)
                                {
                                printf("Invalid command line argument\n\n");
                                return(DosBeep(1000,100));
                                }
                          *p = '\0';
                          rc = CreateServerSession(
&szArg[2],p+1,hList,&dwServerSessions[nSSessions++]);
                          *p = '.';
                          if (rc != SUCCESS)
                                printf("Creating %s session (result = %d),
                                session
(%ld)\n",szArg,rc,dwServerSessions[nSSessions-1]);

                          if (rc != SUCCESS)
                                return(DosBeep(1000,100));
                          }
                    }
              }
```

```
        rc = CreateMessageHandlerList(&hCList);
        if (rc != SUCCESS)
              printf("Creating client message handler list (%d)\n",rc);
        if (rc != SUCCESS)
              return(DosBeep(1000,100));

        rc = InstallMessageHandler(hCList,OP_SERVICE1,ClientHandler1);
        if (rc != SUCCESS)
              printf("Installing client message handler 1 (%d)\n",rc);
        if (rc != SUCCESS)
              return(DosBeep(1000,100));

        rc = RegisterDefaultMessageHandler(hCList,0);
        if (rc != SUCCESS)
              printf("Registering client default message handler 1
  (%d)\n",rc);
        if (rc != SUCCESS)
              return(DosBeep(1000,100));

        rc = RegisterExceptionHandler(hCList,0);
        if (rc != SUCCESS)
              printf("Registering client default exception handler 1
  (%d)\n",rc);
        if (rc != SUCCESS)
              return(DosBeep(1000,100));

        printf("\n\nSystem properly initialized.  Press any key to connect
  ");
        fflush(stdout);
        getchar();
        printf("\n");

        // Connect to remote servers

        for (i = 1; i < argc; i++)
              {
              strcpy(szArg,argv[i]);
              if (szArg[0] == '-')
                    {
                    if (toupper(szArg[1]) == 'C')
```

```
                        {
                        p = (char *) strchr(szArg,'.');
                        if (p == 0)
                              {
                              printf("Invalid command line argument\n\n");
                              return(DosBeep(1000,100));
                              }
                        *p = '\0';
                        strcpy(szClientName,&szArg[2]);
                        rc = ClientConnect(szClientName,p+1,hCList,
                              &dwClientSessions[nCSessions++]);
                        *p = '.';
                        printf("Connecting with server %s (result = %d),
                              session
(%ld)\n",p+1,rc,dwClientSessions[nCSessions-1]);
                        if (rc != SUCCESS)
                              return(DosBeep(1000,100));
                        }
                  }
            }

      j = 0;
      while (1)
            {
            for (i = 0; i < nCSessions; i++)
                  {
                  printf("Sending OP_REQUEST1 to session
%ld...\n",dwClientSessions[i]);
                  rc = DispatchNetMsg(1);
                  rc =
SendNetMessage(dwClientSessions[i],OP_REQUEST1,0,0,0);
                  if (rc != SUCCESS)
                        {
                        while (rc == ERROR_CONNECT_IN_PROCESS)
                              {
                              DispatchNetMsg(1);
                              rc =
SendNetMessage(dwClientSessions[i],OP_REQUEST1,0,0,0);
                              }
                        }
```

```
                    }
                }
        return(0);
}


int MessageHandler1(unsigned long dwSessionId,PMSG pMsg)
{
        int rc;

        printf("OP_REQUEST1 arrived ***  (session %ld)  \n",dwSessionId);
                pMsg->dwOpCode = OP_SERVICE1;
                pMsg->dwSequence = 2;
                pMsg->dwFlags = 0;
                pMsg->dwBuffLen = 0;

        rc = SendNetMessage(dwSessionId,OP_SERVICE1,0,0,0);
        printf("Sending OP_SERVICE1 ***  (%d)\n",rc);
        if (rc != SUCCESS)
                {
                while (rc == ERROR_CONNECT_IN_PROCESS)
                        {
                        DispatchNetMsg(0);
                        rc = SendNetMessage(dwSessionId,OP_SERVICE1,0,0,0);
                        }
                }
        return(rc);
}

int DefMsgHandler(unsigned long dwSessionId,PMSG pMsg)
{
        return(0);
}

int ExceptionHandler(unsigned long dwSessionId,PMSG pMsg)
{
        return(0);
}

int ClientHandler1(unsigned long dwSessionId,PMSG pMsg)
```

```
{
      printf("**** OP_SERVICE1 arrived  (session %ld)\n",dwSessionId);

      return(0);
}
```

The Grant conversation is originated by the Retail Accounting System 1504 of FIG. 15 in the store, most likely as a result of a request to purchase originated by the customer at the PS. The structure of the message-dependent section of the message passed back and forth between these two entities looks as follows:

```
#define OP_GRANT        0x0170      // Command
#define OP_GRANTED      0x0171      // Response
#define OP_DENIED       0x0172      // Response

typedef struct _grantstruct
      {
      unsigned long dwOrderNum;
      unsigned long dwStore;
      unsigned long dwCIT;
      unsigned long dwMedia;
      char szFIlename[SIZEOF_ALBUM_NAME+1];
      char szGrantKey[SIZEOF_GRANT_KEY+1];

      } GRANTSTRUCT, *PGRANTSTRUCT;
```

For example, a message sent from the Retail Accounting System 1504 to the Central Host 1506 could look as follows:

```
GRANT ->

 Op Code    Sequence    Flags       Bufflen     Buffer

 00000170   00000001    FFFFFFFF    00000025    76 54 32 45 54 01 02 12 64
                                                88 ....
```

And the return message (authorization) could look like:

```
GRANTED <-

 Op Code    Sequence    Flags       Bufflen     Buffer

 00000171   00000001    FFFFFF      00000025    54 34 98 76 54 01 90 23 20
                                                ...
```

Where the flags field has an unknown meaning at this point.
Had the request been denied the return message would look like:

```
DENIED <-

 Op Code    Sequence    Flags       Bufflen     Buffer

 00000172   00000001    FFFFFF      00000010    54 09 70 23 41 ...
```

Where the flags field probably describes the reason for the denial.

The Manufacturing conversation is originated by the Retail Accounting System. The structure of the message passed back and forth between the Retail Accounting 1504 and the Manufacturing Control Station 1508 looks as follows:
Note: These two structures are shared by both the Order and Manufacturing conversations.

```
#define OP_MAKE                 0x0110      // Command
#define OP_MADE                 0x0111      // Response
#define OP_CANNOT_MAKE          0x0112      // Response


 typedef struct _itemstruct
      {
      unsigned long dwMedia;
      unsigned long dwCopies;
      unsigned long dwTracks;
      unsigned long dwCDBarcode;
      unsigned long dwPricePerItem;
      char szFileName[SIZEOF_ALBUM_NAME+1];
      char szAlbumTitle[SIZEOF_ALBUM_TITLE+1];
```

```
        } ITEMSTRUCT, *PITEMSTRUCT;

typedef struct _orderstruct
        {
        unsigned long dwOrderNum;
        unsigned long dwCustomerID;
        unsigned long dwCIT;
        unsigned long dwNumLineItems;
        char szHowPaid[SIZEOF_HOW_PAID+1];
        ITEMSTRUCT ItemBuff[1];

        } ORDERSTRUCT, *PORDERSTRUCT;
```

For example, a message sent from the Retail Accounting Station to the Manufacturing Control Application station 1508 could look as follows:

```
MAKE ->

 Op Code    Sequence    Flags       Bufflen     Buffer
 00000110   00000001    FFFFFF      00000011    54 09 70 23 41 ...

MADE <-
 Op Code    Sequence    Flags       Bufflen     Buffer
 00000111   00000001    FFFFFF      00000011    54 09 70 23 41 ...
```

The inability to manufacture would be signaled by a message with the following contents:

```
CANNOT_MAKE <-
 Op Code    Sequence    Flags       Bufflen     Buffer
 00000112   00000001    FFFFFF      00000011    54 09 70 23 41 ...
```

Where the flags field indicates the reason why the requested item cannot be manufactured, and perhaps also suggests a recommended action to remedy the problem. Problems such as device out of service, unable to establish NetBios session, etc, are likely to be the reasons encoded in the flags field of the message.

The Stream conversations are more complex than the others since the conversation occurs in states such as OPEN, READ, CLOSE.

```
#define OP_EXIST          0x0130      // Command
#define OP_EXISTS         0x0131      // Response
#define OP_OPEN           0x0132      // Command
#define OP_OPENED         0x0133      // Response
#define OP_SEEK           0x0134      // Command
#define OP_SEEK_RESULT    0x0135      //Response
#define OP_READ           0x0136      // Command
#define OP_BUFF_READ      0x0137      // Response
#define OP_WRITE          0x0138      // Command
#define OP_WRITTEN        0x0139      // Response
#define OP_BLAST          0x0140      // Command
#define OP_BLASTED        0x0141      // Response
#define OP_XON            0x0142      // Command
#define OP_XOFF           0x0143      // Command
#define OP_CLOSE          0x0144      // Command
#define OP_TRUNCATE       0x0145      // Command
#define OP_TRUNCATED      0x0146      // Response
#define OP_STATUS         0x0147      // Informational (async)

typedef struct _streamstruct
      {
      unsigned long idCIT;
      unsigned char Buffer[1];

      } STREAMSTRUCT, *PSTREAMSTRUCT;
```

For example, a conversation established between the PS and the Data Server or between the Device Controller and the Data Server might look as follows:

```
EXIST ->

 Op Code    Sequence    Flags      Bufflen     Buffer

 00000130   00000001    FFFFFF     00000009    54 09 70 23 41 ...

EXISTS <-
```

Op Code	Sequence	Flags	Bufflen	Buffer
00000131	00000001	FFFFFF	00000010	54 09 70 23 41 ...

OPEN ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000132	00000001	FFFFFF	00000010	54 09 70 23 41 ...

OPENED <-

Op Code	Sequence	Flags	Bufflen	Buffer
00000133	00000001	FFFFFF	00000010	54 09 70 23 41 ...

READ ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000136	00000001	FFFFFF	00000010	54 09 70 23 41 ...

BUFF_READ <-

Op Code	Sequence	Flags	Bufflen	Buffer
00000137	00000001	FFFFFF	00000010	54 09 70 23 41 ...

SEEK ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000134	00000001	FFFFFF	00000010	54 09 70 23 41 ...

SEEK_RESULT <-

Op Code	Sequence	Flags	Bufflen	Buffer
00000135	00000001	FFFFFF	00000010	54 09 70 23 41 ...

BLAST ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000140	00000001	FFFFFF	00000010	54 09 70 23 41 ...

BLASTED <-

Op Code	Sequence	Flags	Bufflen	Buffer
00000141	00000001	FFFFFF	00000010	54 09 70 23 41 ...

XON ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000142	00000001	FFFFFF	00000010	54 09 70 23 41 ...

XOFF ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000143	00000001	FFFFFF	00000010	54 09 70 23 41 ...

TRUNCATE ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000145	00000001	FFFFFF	00000010	54 09 70 23 41 ...

TRUNCATED <-

Op Code	Sequence	Flags	Bufflen	Buffer
00000146	00000001	FFFFFF	00000010	54 09 70 23 41 ...

WRITE ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000138	00000001	FFFFFF	00000010	54 09 70 23 41 ...

WRITTEN <-

Op Code	Sequence	Flags	Bufflen	Buffer
00000139	00000001	FFFFFF	00000010	54 09 70 23 41 ...

CLOSE ->

Op Code	Sequence	Flags	Bufflen	Buffer
00000144	00000001	FFFFFF	00000010	54 09 70 23 41 ...

```
STATUS <->
 Op Code    Sequence    Flags       Bufflen     Buffer
 00000147   00000001    FFFFFF      00000010    54 09 70 23 41 ...
```

The Order conversation can exist between the PS 1502 and the Retail Accounting Station 1504.

```
#define OP_ORDER              0x0160      // Command

typedef struct _orderstruct
      {
      char  szileName[9];

      } ORDERSTRUCT, *PORDERSTRUCT;
```

For example, a message sent from the PS 1502 to the Retail Accounting Station 1504 during the process of placing an order could look as follows:

```
ORDER ->
Op code  Sequence  Flags  Bufflen  Buffer
00000160 00000001  FFFFFF 00000009 FDS453FG0
```

The Device conversation takes place within the Manufacturing Control station 1508.

```
#define OP_PREPARE      0x0180      // Command
#define OP_READY        0x0181      // Response
#define OP_START        0x0182      // Command
#define OP_STARTED      0x0183      // Response
#define OP_RESTART      0x0184      // Command
#define OP_RESTARTED    0x0185          // Response
#define OP_ABORT        0x0186      // Command
#define OP_ABORTED      0x0187      // Response
#define OP_PAUSE        0x0188      // Command
#define OP_PAUSED       0x0189      // Response
#define OP_DEVSTATUS    0x0190      // Informational (async)
#define OP_COMPLETED    0x0193      // Informational (async)
#define OP_SHUTDOWN     0x0194      // Command
```

```
#define OP_DOWN          0x0195      // Response

typedef struct _devicestruct
      {
      unsigned long dwStatusCode;   // return status codes
      unsigned long dwDeviceType;   // device type
      unsigned long dwTrackCount;   // track (song) count
      unsigned char szFileName[12]; // audio/image file name
      unsigned char szExtraBuf[1];  // pointer to extra data

      } DEVICESTRUCT, *PDEVICESTRUCT;
```

For example, a typical conversation between the Manufacturing Control Application at station 1508 and a particular Device Controller may develop as follows:

```
OP_PREPARE ->
 Op Code    Sequence    Flags       Bufflen     Buffer
 00000180   00000001    FFFFFF      00000010    54 09 70 23 41 ...

OP_READY <-
 Op Code    Sequence    Flags       Bufflen     Buffer
 00000181   00000001    FFFFFF      00000010    54 09 70 23 41 ...

OP_START ->
 Op Code    Sequence    Flags       Bufflen     Buffer
 00000182   00000001    FFFFFF      00000010    54 09 70 23 41 ...

OP_STARTED <-
 Op Code    Sequence    Flags       Bufflen     Buffer
 00000183   00000001    FFFFFF      00000010    54 09 70 23 41 ...

 OP_RESTART ->
 Op Code    Sequence    Flags       Bufflen     Buffer
```

```
00000184  00000001  FFFFFF  00000010  54 09 70 23 41 ...

OP_RESTARTED <-

Op Code   Sequence  Flags   Bufflen   Buffer

00000185  00000001  FFFFFF  00000010  54 09 70 23 41 ...

OP_ABORT ->

Op Code   Sequence  Flags   Bufflen   Buffer

00000186  00000001  FFFFFF  00000010  54 09 70 23 41 ...

OP_ABORTED <-

Op Code   Sequence  Flags   Bufflen   Buffer

00000187  00000001  FFFFFF  00000010  54 09 70 23 41 ...

OP_PAUSE ->

Op Code   Sequence  Flags   Bufflen   Buffer

00000188  00000001  FFFFFF  00000010  54 09 70 23 41 ...

OP_PAUSED <-

Op Code   Sequence  Flags   Bufflen   Buffer

00000189  00000001  FFFFFF  00000010  54 09 70 23 41 ...

OP_DEVSTATUS <-

Op Code   Sequence  Flags   Bufflen   Buffer

00000190  00000001  FFFFFF  00000010  54 09 70 23 41 ...
OP_COMPLETED <-

Op Code   Sequence  Flags   Bufflen   Buffer

00000193  00000001  FFFFFF  00000010  54 09 70 23 41 ...

OP_SHUTDOWN ->

Op Code   Sequence  Flags   Bufflen   Buffer

00000194  00000001  FFFFFF  00000010  54 09 70 23 41 ...
```

```
OP_DOWN <-

 Op Code    Sequence    Flags       Bufflen     Buffer

 00000195   00000001    FFFFFF      00000010    54 09 70 23 41 ...
```

In FIG. 16 there is illustrated a panorama of multimedia preview stations or booths 1303 as previously illustrated for FIGS.12 and 13. These booths 1303 are similar in concept to the multimedia preview booths 106 of FIG.1 and are those booths illustrated in FIG.11 at the retail store 1108. FIG. 16 illustrates a typical video and audio system including a monitor touch screen and controller 1606 for video, and an audio system including speakers 1608 and a sound amplifier 1610. A system unit 1612 provides for interconnect with the selected ATM or LAN adapter and further includes a MPEG decoder and sound decompressor with a communication network connection at network unit 1614. At the completion of the customer's initial preview and ordering transaction, the slip printer 1602 provides a hard copy record of the customer's order.

A multimedia preview station 1303 similar to the station 1303 of FIG. 16 is illustrated in FIG. 17. The alternative of FIG.17 provides for a stereo television unit 1702, which differentiates it from FIG.16 as does the inclusion of a set-top video decoder unit 1704. The individualized arrangement of FIG. 17, including the stereo television 1702 and in particular the set-top video decoder 1704, emphasizes the potential of personalized customer premise arrangements which can be provided for outside of the retail store environment. Further, such "home" units can include their own built in ATM hardware and software interfaces. An additional example is a color printer which could be equipped with a direct communication unit providing a connect to a communications data network which eliminates the need for a SCSI bus attachment or an interface card on the computer bus as is typical for such printing units.

The present invention as presented herein is adaptable by those of ordinary skill in the art to accommodate customer requests which originate from any customer premise, including a customer's residence. The preview station 1303 of FIG. 17 is not significantly dissimilar from a home entertainment center equipped with stereo television and a set-top video decoder with cable television service from such providers as COMCAST, TCI, SELKIRK, LEADERSHIP and other similar companies. The customer or home consumer can through the set-top unit preview, for example, authorized movies or other digital data, including video games and experience the requested feature communicated from a remotely disposed data server. Alternatively, the consumer can through the set-top unit place an order for the item of interest either identifying it for later pick-up or home delivery within a short period of time.

The preview and order process at the customer premise was generally introduced with regard to FIG. 13. There it was identified that the customer, through the touch screen panel further illustrated in FIGS. 18 through 28, initiated the data preview and content data ordering through touch manipulation of the touch pads at the touch screen monitor.

FIG. 18 is one of the initial screens which will be presented to the customer. This library screen shows by way of example six touch pads with descriptive legends identifying selection categories such as MUSIC, MOVIES/VIDEOS, VIDEO GAMES, AUDIO BOOKS, COMPUTER SOFTWARE and MULTIMEDIA. Once a touch pad is contacted, the program provides for a visual indicator, such as a change in color of the pad, that the selection has been received and is understood. In the case of FIG. 18, if the MUSIC pad is touched the category screen as illustrated in FIG. 19 will be presented.

The category screen will at the top of the screen identify the customer by name as a result of the customer having presented the customer's membership or other identifying card and having inserted it into the card reader receptacle slot 1604 of FIG. 16 for initiating the transaction process at the Preview station 1303. Shown by way of example in FIG. 19, are 12 categories of music which are identified as touch pads for POP/ROCK, COUNTRY, BLUES, R&B/SOUL, EASY LISTENING, RAP, JAZZ, RELIGIOUS, SOUND TRACK, CLASSICAL, NEW AGE and COMEDY.

Along the right margin of the screen are five transactional touch pads to further assist the customer in the selection process. The top-most touch pad with the legend "Search" if touched will present the screen of FIG. 21 which will be discussed subsequently. The second from the top touch pad with the legend "Previous Screen," in the case of the present example, would return the customer to the screen in FIG. 18. The middle touch pad with the legend "Shopping Cart" will advance the screen presentation to the screen identified in FIG. 27 which will be discussed hereinafter. The touch pad second from the bottom has a legend identified as "Help." When the help touch pad is initiated, video instructions with respect to the touched screen will appear to aid the customer in the use of the screen, in this case for FIG. 19. If the bottom-most touch pad with the legend "Quit" is energized, the screen will be switched back to one of the most initial screens, not discussed

here, which presents an attraction screen to encourage a walk-by customer to participate in the preview and ordering process unless an order has been made by placing an item in the Shopping Cart, in which case, the screen in FIG. 28 will be displayed.

The screen in FIG. 20 is what will be presented to the customer if the POP/ ROCK touch pad of FIG. 19 is initiated. It provides to the customer an alphabetic listing of artists and groups included within the POP/ROCK category. At the right margin of the screen near the top is a triangular shaped touch pad which allows a customer to page up for artists or groups which may be listed at the near end of the alphabet, whereas an inverted triangle at the bottom right hand margin provides a touch pad for allowing the customer to page down to the far end of the alphabetical listing.

In addition, by touching the vertically disposed alphabet letters presented at the right side of the screen, the customer can by touching a particular letter of the alphabet have presented on the screen a list of artists or groups, which list starts with the first artist or group in the selected letter listing. The three assistance touch pads at the bottom include the "Help" touch pad which if touched will present a screen which explains the operation of the screen for FIG. 20. If the touch pad with the legend "Cancel" is touched, then in this exemplary case the category screen of either FIG. 19 or the keyboard screen of FIG. 21 will be presented dependent upon the path which the customer utilized in reaching this screen 20. If in the alternative, the customer touches the pad with the legend "OK", the screen will change to the screen of FIG. 23 which will show the multiple album listings of the selected artists or in the case of an artist who has only a single album, the next screen will be the screen identifying that particular single album as in FIG. 24. It should be noted that in FIG. 20 at the very top there is a vivid border outline around the first listed artist identifying that artist as the customer's selection. To change the selection, the customer must touch the name of another artist or group.

The screen of FIG. 21 is the keyboard screen which presents a typical typewriter pad for inquiry initiation by letter and number of a description of any of the subjects of the four pads identified centrally in the screen by the legends "Find Song," "Find Artist," "Find Album," and "Find Label." If for example the customer types by touching in the letters "JO" and the "Touch Artist" pad is initiated, the next screen that will be presented will be the screen of FIG. 22 which identifies all such artists or groups with a name starting with "JO". In FIG. 21, as explained elsewhere herein, the "Help" touch pad will present a screen for the customer's aid in providing an explanation of the manner of use of the screen of FIG. 21. The "Cancel" touch pad at the very right hand corner when touched will, in this instance, return the viewer to the category screen of FIG. 19 or possibly FIG. 23 or 24 if the touch pad with the legend "Search" was previously touched at these respective screens.

Similarly, entering search criteria on the screeen depicted in FIG. 21 and touching the "Find Song" or "Find Album" or "Find Label" pad, will result in a screen similar to that depicted in FIG. 22 with a list of the respective search results, such as a list of song titles or album names or albums of a specific music recording label depending on the pad pressed.

In the presented example, FIG. 22 illustrates a list of artists. The vivid border around the listed artist indicates the selection the customer has made. If the customer touches the touch pad with the legend "OK" the customer will be presented with the screen of FIG. 23. The touch pads at the right border of the screen of FIG. 23 are exemplary and are discussed in general with respect to other figures and screens. Selecting a specific album by touching the album picture or the screen causes the screen in FIG. 24 to be presented.

The customer when presented with the screen of FIG. 24 which lists a specific album title, artist and label company for that particular album, has the choice of selecting, from the listed pieces on that album, both audio and video as indicated to the right side of each of the listed pieces where audio is identified as being available if the headphone set icon 2402 is presented and video is identified as being available if the television set icon 2404 is present. The headset icon 2402 and the TV icon 2404 are touch points which can be touched to initiate preview of the selected song or music video.

The screen in FIG. 24 is identified as the song screen, and if there are more selections than can be presented on an individual screen, then the triangular shaped touch points are available for paging up or paging down. Further, the touch points or pads at the right margin of the figure are exemplary of past discussed touch pads with similar legends. At the bottom of the screen, there are three touch points with legends, including "Place CD in Cart," "Place Cassette in Cart" and "Suggest." If the CD touch point is touched, then the particular listed album will be presented to the customer in the customer's shopping cart as a CD. Likewise, the customer may select a cassette if the cassette touch pad is touched. If either of the touch pads are consecutively touched, then each time the touch pad is contacted it will place an additional CD or cassette in the shopping cart. The "Suggest" touch point of FIG. 24 when touched will present additional selections similar to the artist or group identified at the top portion of the screen 24 which can be tailored to the customer's interest based upon information available from the Customer Data Base 128 of FIG.1.

When a headset icon 2402 is touched, such as those illustrated in FIG. 24, then a screen such as that illustrated in FIG. 25 which is an audio screen will be presented for preview of the identified song or piece by

the artist. It will be noted that the audio screen for preview is superimposed upon the screen which was identified for FIG. 24. Only the touch pads of the superimposed screen are active when presented with the screen of FIG. 25. Presented at the bottom portion of the superimposed screen of FIG. 25 are two touch pads. The left most pad being a touch pad to stop and cancel the preview presentation, whereas the right most touch pad is a "Pause" touch pad for momentary stopping the audio presentation. At the right side of the superimposed screen is a double ended vertical pointer which provides a volume control which is dependent upon the level of the point touched or manipulated by the customer.

In FIG. 26 the customer is presented with a presentation of the video selection if the customer has initiated a touch at the TV icon 2404 as presented in FIG. 24. The screen of 26 has similar "Stop and Cancel," "Pause" and volume touch point controls as in FIG 25.

FIG. 27 presents a screen which is a Shopping Cart screen which identifies those album selections which the customer has made during the course of the customer's presence in the preview booth 1303. It presents the customer with the quantity, the title, the format, i.e., CD or cassette, itemized with subtotal, and tax and total prices. The touch pads at the bottom with the legends "Quantity," "CD," "Cassette" and "Delete" allow the customer to change the quantity or change the format, or to delete selections if there should be a change of mind on the part of the customer or the customer needs to make a correction in the order. The touch pad with the legend "Quantity" can increase the quantity by touching at the top part of the diamond shaped touch pad or by touching at the bottom to reduce the quantity.

In FIG. 28 the customer is presented with this screen to place an order or in the event that during the course of the preview and ordering, the customer has requested to discontinue the ordering process and has touched the pad with the legend "Quit" when the shopping cart of the customer at that particular time has identified items for purchase. A video screen will appear with a sales clerk audibly presenting a statement to the customer, as shown in print at the bottom of the screen, suggesting that the customer review the customer's shopping cart and make a determination, based on the touch pads at the right margin, to clarify the order status and the purchase and order intentions of the customer.

The on-demand digital data reproduction system of the present invention provides numerous accounting tools which have flexibility in their location placement, and, for example, can reside at the local retail store level as shown at retail store 110 in FIG.1, in particular at the accounting management subsystem 108, and for example the retail accounting and management subsystem 1306 of FIG.13 and also that which is shown, by way of example, in FIG.15 at the retail accounting manager's workstation 1504.

A few of the transactional analysis tool workstation screens of the manager's workstation are illustrated in FIGS.29-36, with a central host analysis tool screen presentation shown in FIG.37. A hard copy sales by-category report in FIG.38, is the result of activity and requests made at the analysis central host workstation in FIG.37.

FIG.29 presents for the operator numerous report opportunities with regard to sales and customer demographics. In the sales category, reports on gross revenue, top selling albums for a category, top selling albums, music category and unit sales by media type can be accessed by way of example. Customer demographics can be further analyzed by reports specific to customer transactions by age, time of day and money spent. Activating the gross revenue sales report will result in the presentation of the screen of FIG.30, where by way of illustration, a report is provided for compact discs and tape sales in monetary amounts. Activating the top selling albums category will result in the screen presentation as shown in FIG.31 where the albums and the specific unit sales count number are indicated. As shown in FIG.31 a hard copy report can be obtained through manipulation of the "Print" touch pad.

At FIG.29, activation of the top selling albums feature will result in presentation of the screen shown in FIG.32 where the albums for all categories are listed as well as the numerical sales count. If the music category analysis tool of FIG. 29 is activated, then the screen presentation would be that of FIG.33 where the categories are illustrated, and in this particular example, in a pie chart formation. The touch pad to the right of the screen in FIG. 33 is for other graphical selections such as bar graphs or a print in hard copy. Similar to the presentation in FIG.33 is the presentation of FIG.34 where the data format, such as for CDs and tapes, can be illustrated in the pie chart graphical presentation. The screen of FIG.34 can be obtained by activation of the sales category by media type at the screen of FIG.29.

For customer demographics the activation of customer transactions by age in FIG.29 results in the presentation of the screen in FIG.35, whereas, the time of day for a transactions count, when activated, presents the screen of FIG.36. Likewise, the money spent category in FIG.29 pulls up and displays screens similar to those of FIGS 35 and 36 for money amounts spent as a function of time at an identified particular store operation.

The screen of FIG. 37 is a higher level analysis tool for the central host workstation such as that illustrated in FIG.11 for the central repository and accounting subsystem 1110 or in FIG.15 for the central host accounting

1506. Transactional operation at this screen can provide significant managerial information and data with regard to the sales performance and accountability by retail store, by geographic regions or by a chain of particular stores and the hard copy presentation of FIG.38 shows, by way of example, particular sales performance aspects of that available data.

While the invention has been particularly shown and described with reference to a detailed description, it will be understood by those skilled in the art that various changes in form and detail may be made therein without departing from the spirit and scope of the invention.

Claims

1. A method for accessing digital information stored in a source library at a first location and recording a portion of the digital information, which information is processed for subsequent playback, onto a recordable medium at a location other than the first location, the method including the steps of:
identifying, at a second location remote from the first location, a portion of the digital information in the source library;
sending a request from the second location to the source library for the identified portion of the digital information in the source library to be transferred from the source library;
validating that the identified portion of the digital information is authorised for playback;
transferring in real-time information within the identified portion of the digital information to a manufacturing device at a location other than the first location for incorporation of the digital information on a medium suitable for playback; and
incorporating the identified portion of the digital information on the medium in real-time in a manner suitable for playback.

2. A method according to claim 1, including logging the validation step of the identified portion of the digital information in an accounting database.

3. A method according to claim 1 or claim 2, wherein information transferred from the source library is first transferred to the second location and then information is selected from said transferred information for transferring onwards to the manufacturing device for incorporation on a medium suitable for playback.

4. A method according to claim 3, including logging the selection step in an accounting database.

5. A method according to any one of the preceding claims, wherein the manufacturing device to which the information is transferred is located at said second location.

6. A method according to any one of claims 1 to 4, wherein the step of transferring in real-time information within the identified portion of the digital information, transfers the identified portion of the digital information to the manufacturing device at a third location remotely disposed from the first location.

7. A method according to any one of the preceding claims, wherein the step of transferring in real-time information within the identified portion of the digital information transfers the digital information at a high speed constant bit rate.

8. A method according to Claim 5, wherein the step of transferring in real-time the identified portion of the digital information transfers the digital information at a high speed constant bit rate in asynchronous transfer mode.

9. A method according to claim 7, wherein the step of transferring in real-time the identified portion of the digital information transfers the digital information at a high speed constant bit rate which exceeds 44.7 megabits per second.

10. A method according to any one of the preceding claims, where the digital information is stored in a source library at a first location as encrypted and compressed digital information, the method further comprising the step of decrypting and decompressing the selected portion of the digital information, and wherein the step of providing the manufacturing device with the selected portion of the digital information for incorporation of the digital information on the medium provides the decrypted and decompressed selected portion of the digital information.

11. A method according to any one of the preceding claims wherein audio and video digital information is stored in the source library.

12. A method of allocating for storage 100 percentage of content data amongst a plurality of data servers remotely disposed to each other and connected in hierarchal levels having a low and high end and in data communication with a data accessing station also remotely disposed from each of the plurality of data servers, but less so disposed from the data server at the low end of the hierarchy than from each succeeding data server at the next succeeding higher level of the hierarchy through and to the data server at the high end of the hierarchy of data servers, comprising the steps of:

allocating for storage X percentage of the content data at the data server at the low end of the hierarchy least remote from the data accessing station which X percentage of allocated content data is a function of content data for most frequent accessing by the data accessing station;

allocating for storage Y percentage, where Y is < 100 and > X, of the content data at the data server at the high end of the hierarchy of data servers most remote from the data accessing station as a function of content data for least frequent accessing by the data accessing station; and

allocating for storage 100-X-Y percentage, of content data between any other data servers disposed between the low end and the high end data servers, allocating for each succeeding data server more remote from the data accessing station a greater percentage of content data for storage than an allocation for any data server less remotely disposed from the data accessing station and a lesser percentage of content data than the next succeeding server more remotely disposed from the data accessing station.

13. A method according to claim 1, including:

storing digital audio information and digital video information in a source library at a first location, which audio information when originally recorded was for playback at X rate and which video information when originally recorded was for playback at Y rate;

following the steps of validating that the identified portion of the information is authorized for playback, transferring the identified portion of the digital information at a communication rate appropriate for real-time playback at the second location;

providing the manufacturing device with information within the identified portion of the digital information at a communication rate appropriate for direct incorporation of the digital information on the medium for playback of the digital information at X rate for audio information and at Y rate for video information; and

incorporating the selected portion of the digital information on the medium in a manner for playback of audio and video information at the recorded rate.

14. A data communication system for transmitting digital data between remotely located system stations where each station is adapted to participate in a process of storing, accounting for and manufacturing media with digitized content data packaged for playback, comprising:

a point of access station for identifying selected digitized content data and for generating a communication requesting said content data;

a content data storage station having digitized content data remotely disposed from the point of access station;

an accounting station for recording transactional data for each customer point of access station communication requesting digitized content data from the content data storage station;

a manufacturing station remotely disposed from said content data storage station for generating media incorporating the digitized content data requested by the point of access station from the content data storage station; and

a communication network for interconnecting the remotely disposed content data storage station with the point of access station, the accounting station and the manufacturing station to transfer digitized content data in real-time between the content data storage station and the manufacturing station.

15. A data communications system according to claim 14, wherein the communication network for interconnecting the system stations transfers the digitized content data at a high speed constant bit rate from the content data storage station to the manufacturing station.

16. A data communications system according to claim 14 or claim 15, wherein the content data storage station includes:

a master server system station for performing a plurality of server functions, including a first master unit interconnecting a plurality of master units and managing data communications within the master server system and between the master server system station and remotely located system stations, a second master unit for routing data request messages, a third master unit remotely disposed from the point of access station for serving content data stored for fulfilling content data requests made by a consumer at the customer point of access, a fourth master unit for storing customer demographic information for promoting consumer interest in content data and a fifth master unit for managing accounting data as a function of consumer transactions involving consumer requests for content data;

said manufacturing station being remotely disposed from the third master unit;

and said communication network interconnecting the master server system's first master unit with the point of access station and the manufacturing station to transfer in real-time digitized content data between the third master unit for serving stored content data and the manufacturing station.

17. A multimedia preview station including an audio system, a touchscreen monitor and a controller for individually presenting a plurality of screens on the monitor for touch control by a consumer to present to the consumer, consumer identified and requested screen presentations, comprising;

a library screen identifying a plurality of categories of content data for screen presentation on the monitor including a touch point for activating the presentation of a category;

an alphabetical listing screen identifying content data for presentation by alphabetical listing of the category subject matter of the library screen including a touch point for activation by the consumer;

a search screen including monitor touch points for identifying a consumer request either alphabetically or numerically;

a preview screen including monitor touch points for previewing consumer identified audio and video listed content data transferred to the preview screen in real-time and monitor touch points for selecting for purchase listed content data; and

a purchase selection screen including purchase selections made by the consumer, listed by item selected, quantity, format and price, including monitor touch points for altering the item selection.

18. An accounting system including a touchscreen monitor and a controller for individually presenting a plurality of screens on the monitor for touch control by an operator to present to the operator, accounting worktools for analysis of remotely stored digitized data of consumer purchases as a function of sales operation, quantity, monetary amount, purchase format, purchase subject matter and consumer demographics, comprising:

a report screeen identifying a plurality of analysis reports for screen presentation on the monitor including a touchpoint for activating the presentation of an analysis report as a function of consumer sales, consumer demographics, sales unit, and sales group;

a digitized data storage unit remotely disposed from the report screen for transferring at a high speed constant bit rate the digitized data corresponding to a touchpoint activated analysis report on the report screen of the touchscreen monitor; and

an analysis report screen presenting in quantified format the remotely stored digital data corresponding to consumer purchases.

19. A manufacturing system for manufacturing media with digital data content including a plurality of data servers storing digital data content remotely disposed to each other and connected in hierarchal levels comprising:

top level servers each storing a high percentage of the stored digital data content;

intermediate level servers each storing a lesser percentage of stored digital data content than the top level servers;

a high level communication network for interconnecting the top level servers and the intermediate level servers and for transferring digital data content between the top level servers and the intermediate level servers in asynchronous transfer mode;

end site servers each storing a lesser percentage of stored digital data content than the intermediate level servers;

a low level communication network for interconnecting the intermediate level servers and the end site servers and for transferring digital data content between the intermediate level servers and the end site servers in asynchronous transfer mode; and

a manufacturing device interconnected to one of the end site servers for receiving in asynchronous transfer mode digital data content for manufacturing media with the received digital data content.

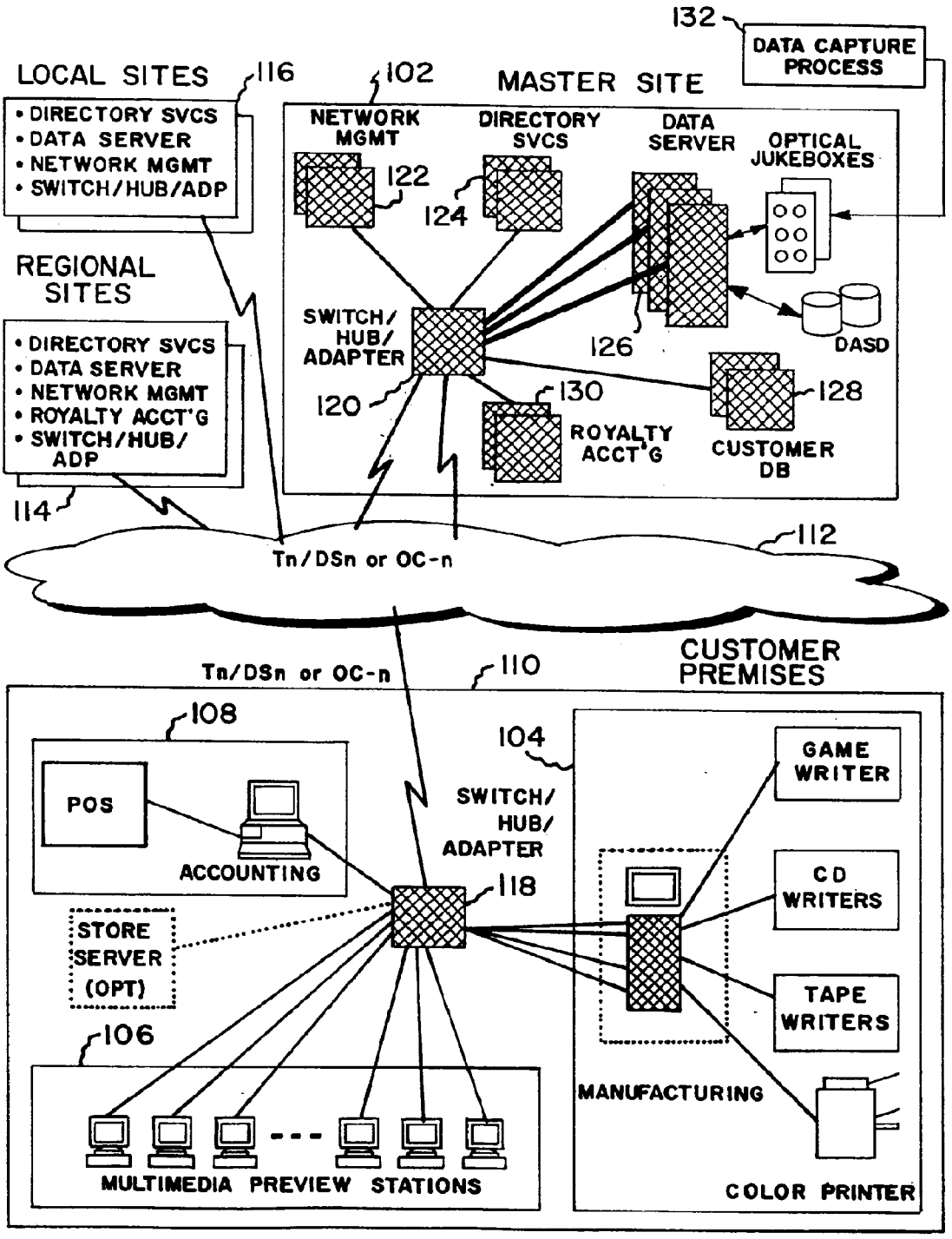

132
DATA CAPTURE PROCESS
LOCAL SITES
116
• DIRECTORY SVCS
• DATA SERVER
• NETWORK MGMT
• SWITCH/HUB/ADP
102
MASTER SITE
NETWORK MGMT
DIRECTORY SVCS
DATA SERVER
OPTICAL JUKEBOXES
122
124
REGIONAL SITES
• DIRECTORY SVCS
• DATA SERVER
• NETWORK MGMT
• ROYALTY ACCT'G
• SWITCH/HUB/ ADP
SWITCH/ HUB/ ADAPTER
126
DASD
120
130
128
ROYALTY ACCT'G
CUSTOMER DB
114
112
Tn/DSn or OC-n
Tn/DSn or OC-n
110
CUSTOMER PREMISES
108
104
POS
ACCOUNTING
SWITCH/ HUB/ ADAPTER
GAME WRITER
CD WRITERS
118
STORE SERVER (OPT)
TAPE WRITERS
106
MANUFACTURING
MULTIMEDIA PREVIEW STATIONS
COLOR PRINTER

FIG. 1

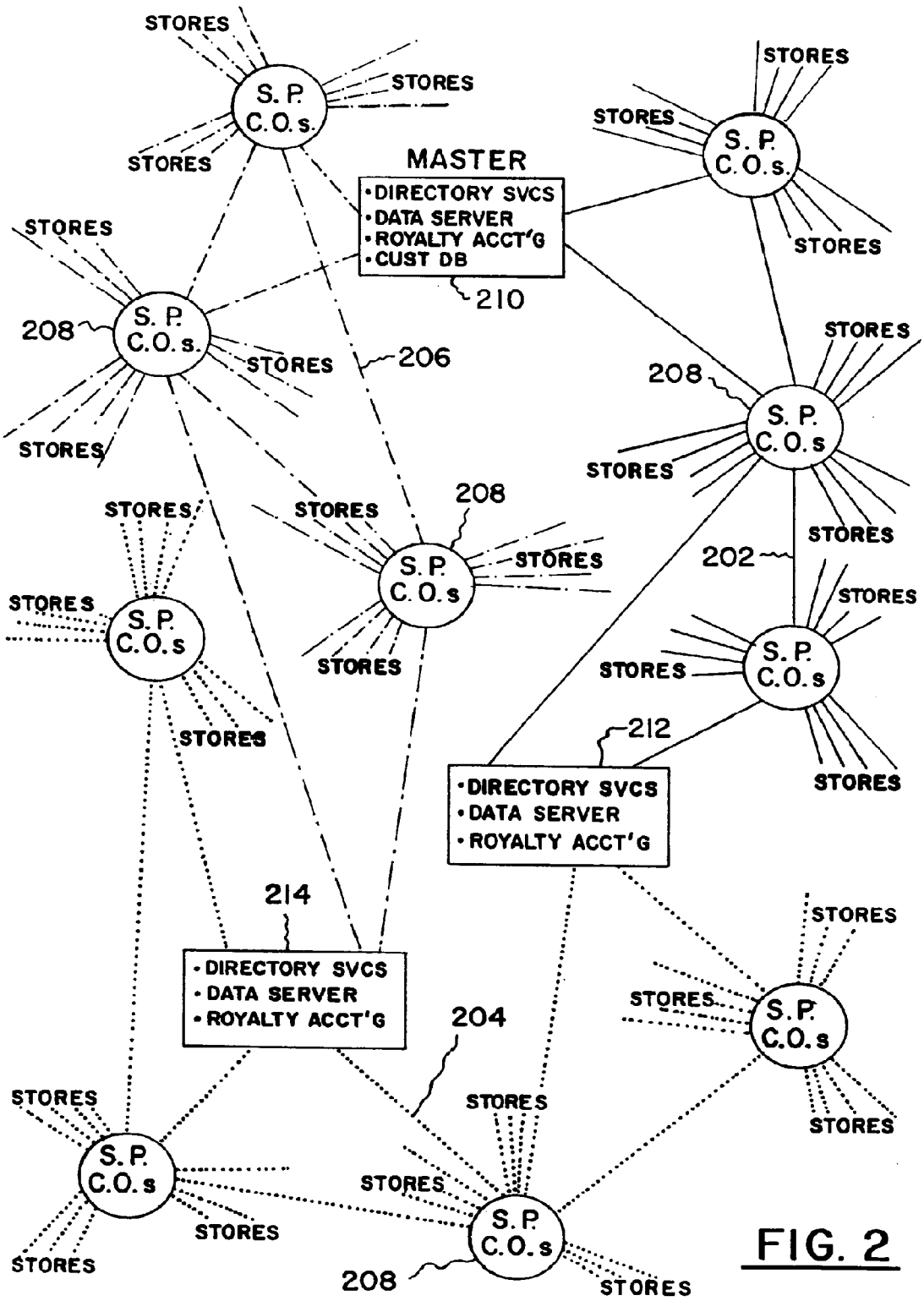

STORES
S. P. C. O. s.
MASTER
• DIRECTORY SVCS
• DATA SERVER
• ROYALTY ACCT'G
• CUST DB
210
208
206
202
212
214
204
• DIRECTORY SVCS
• DATA SERVER
• ROYALTY ACCT'G
S. P. C. O. s

FIG. 2

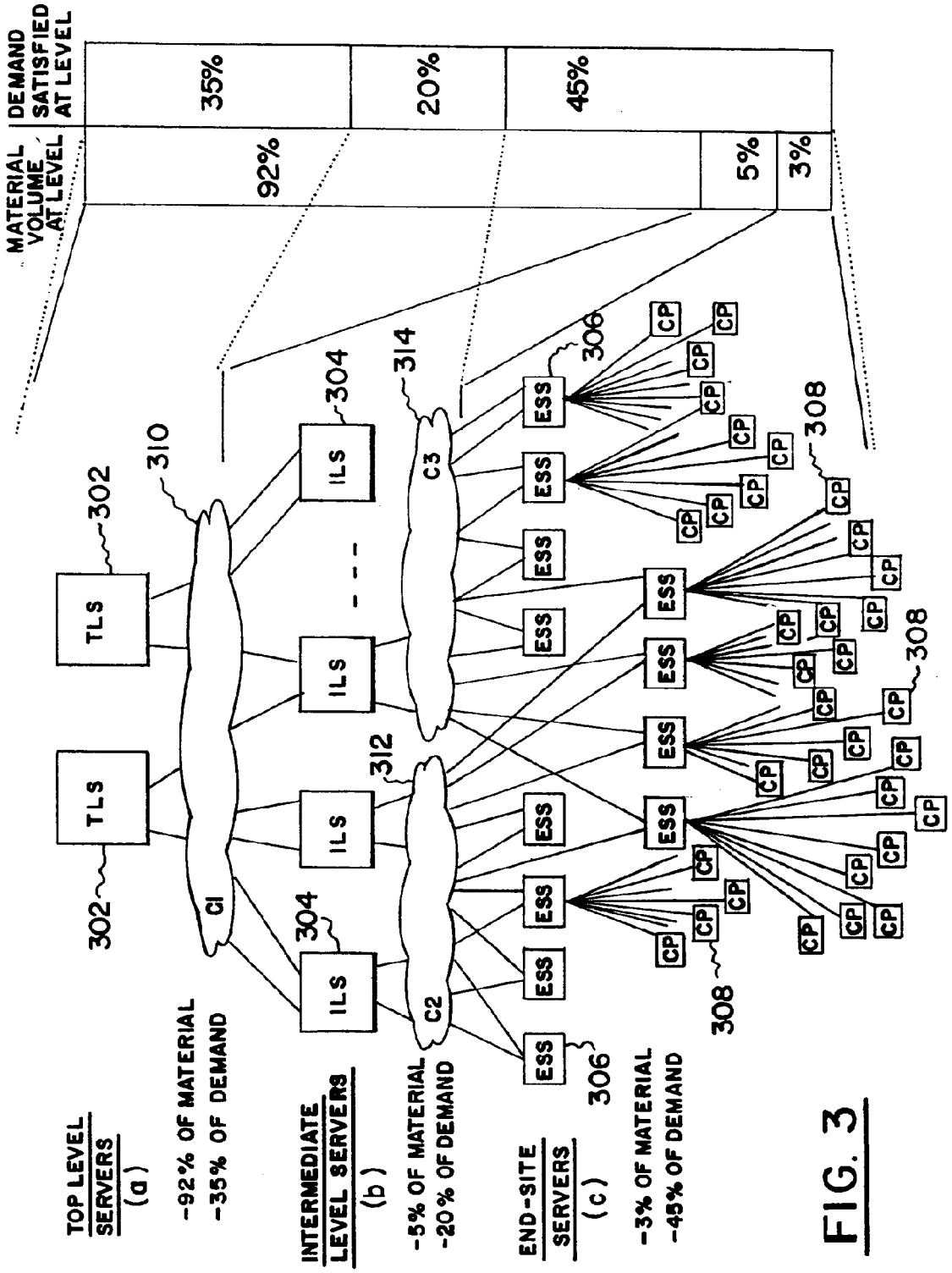

DEMAND SATISFIED AT LEVEL
MATERIAL VOLUME AT LEVEL
35%
20%
45%
92%
5%
3%
TLS
302
310
ILS
304
314
C3
C1
C2
312
ESS
306
CP
308
TOP LEVEL SERVERS (a)
-92% OF MATERIAL
-35% OF DEMAND
INTERMEDIATE LEVEL SERVERS (b)
-5% OF MATERIAL
-20% OF DEMAND
END-SITE SERVERS (c)
-3% OF MATERIAL
-45% OF DEMAND

FIG. 3

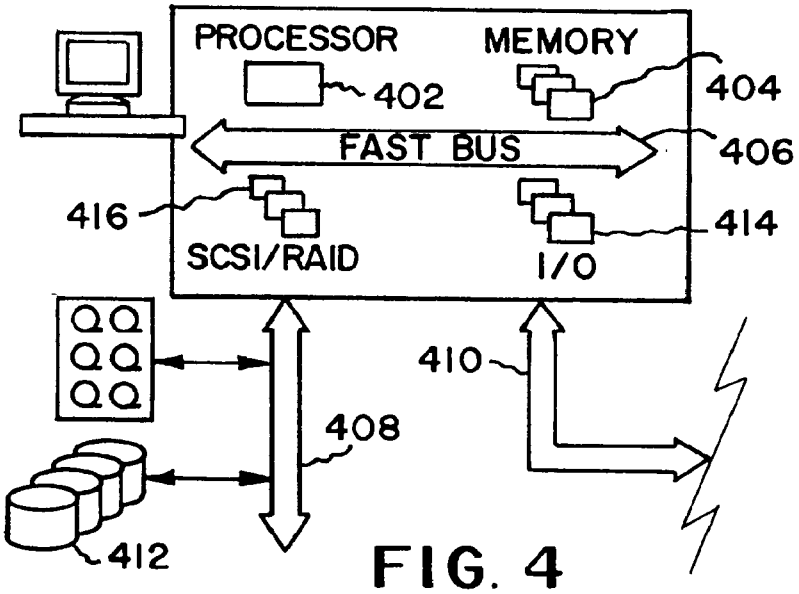

PROCESSOR
MEMORY
402
404
FAST BUS
406
416
SCSI/RAID
I/O
414
410
408
412

FIG. 4

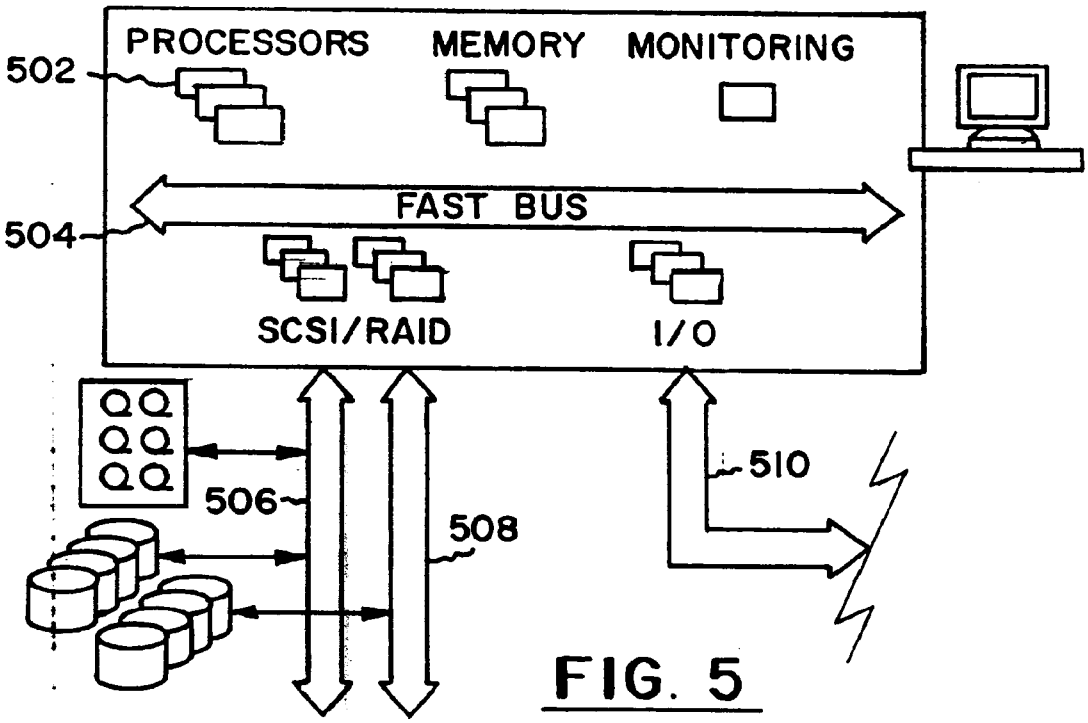

PROCESSORS
MEMORY
MONITORING
502
FAST BUS
504
SCSI/RAID
I/O
510
506
508

FIG. 5

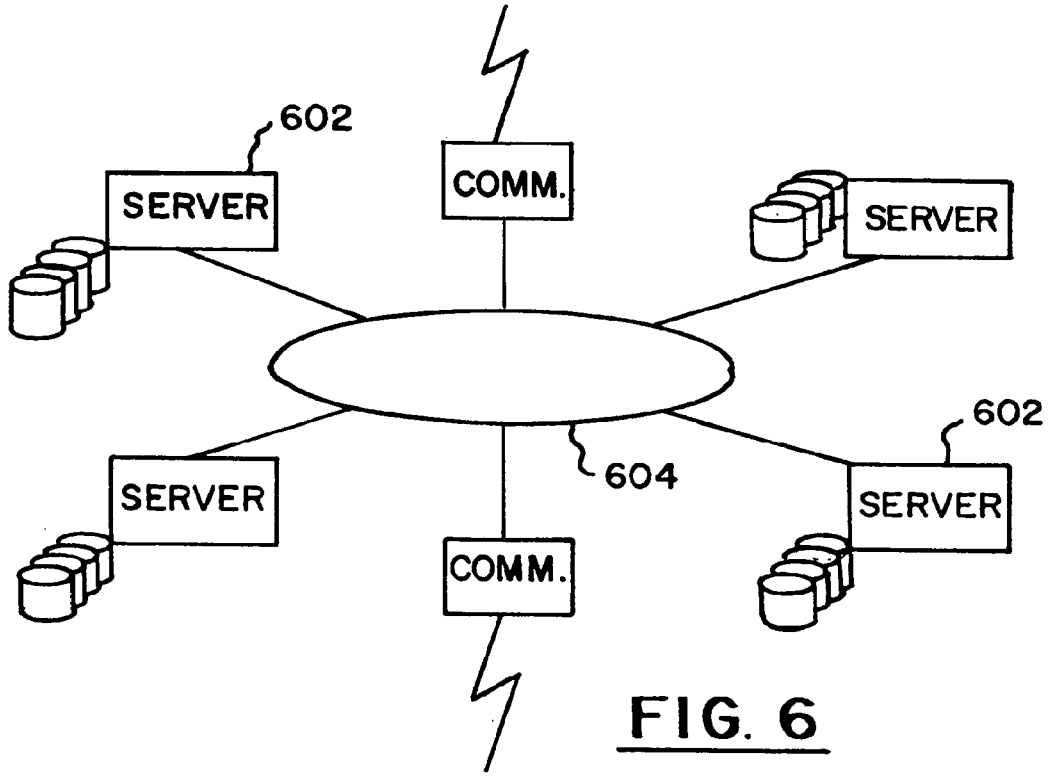

602
SERVER
COMM.
SERVER
602
604
SERVER
SERVER
COMM.

FIG. 6

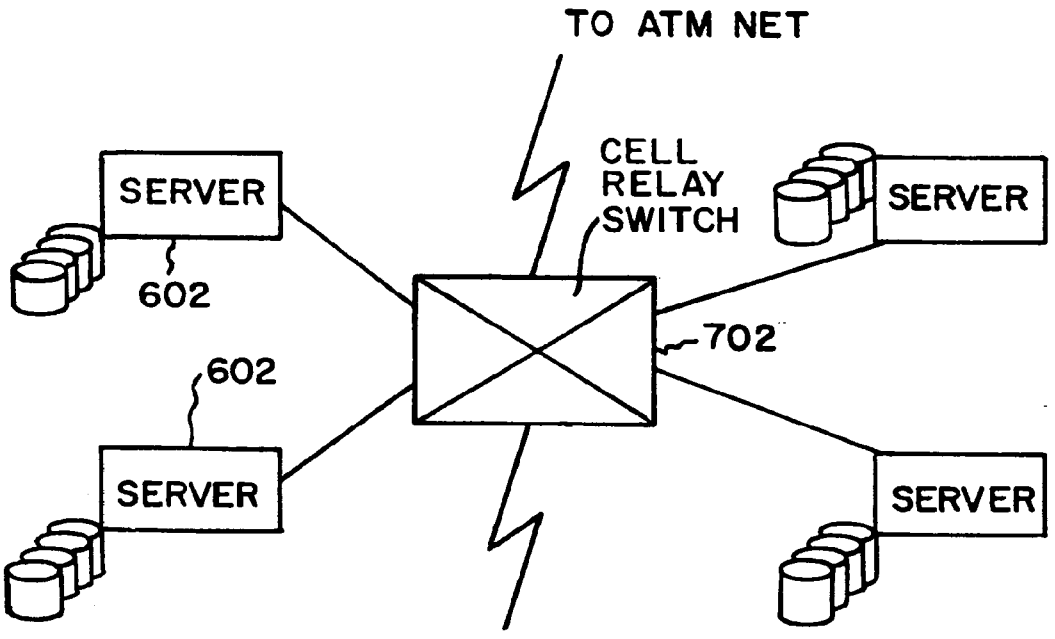

TO ATM NET
CELL
RELAY
SWITCH
SERVER
SERVER
602
702
602
SERVER
SERVER

FIG. 7

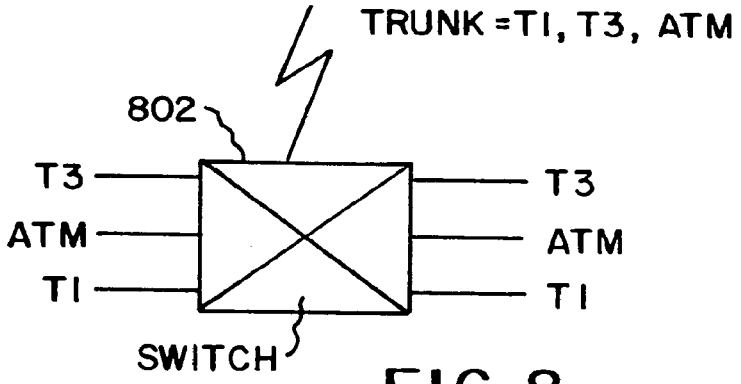


FIG. 8

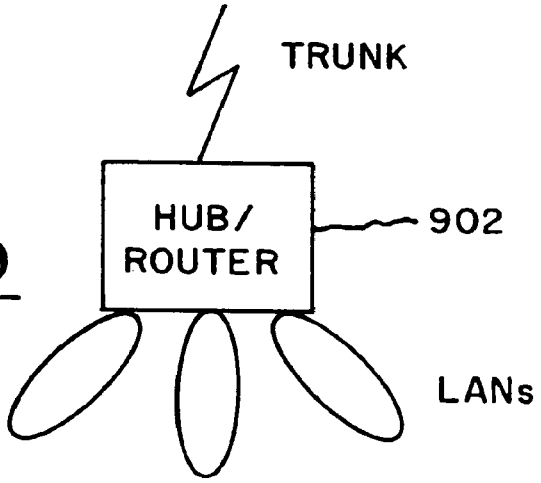


FIG. 9

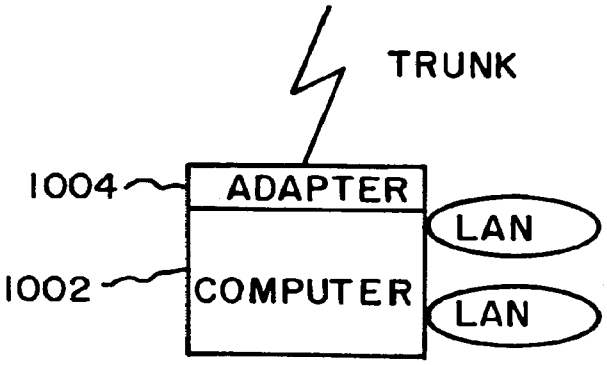


FIG. 10

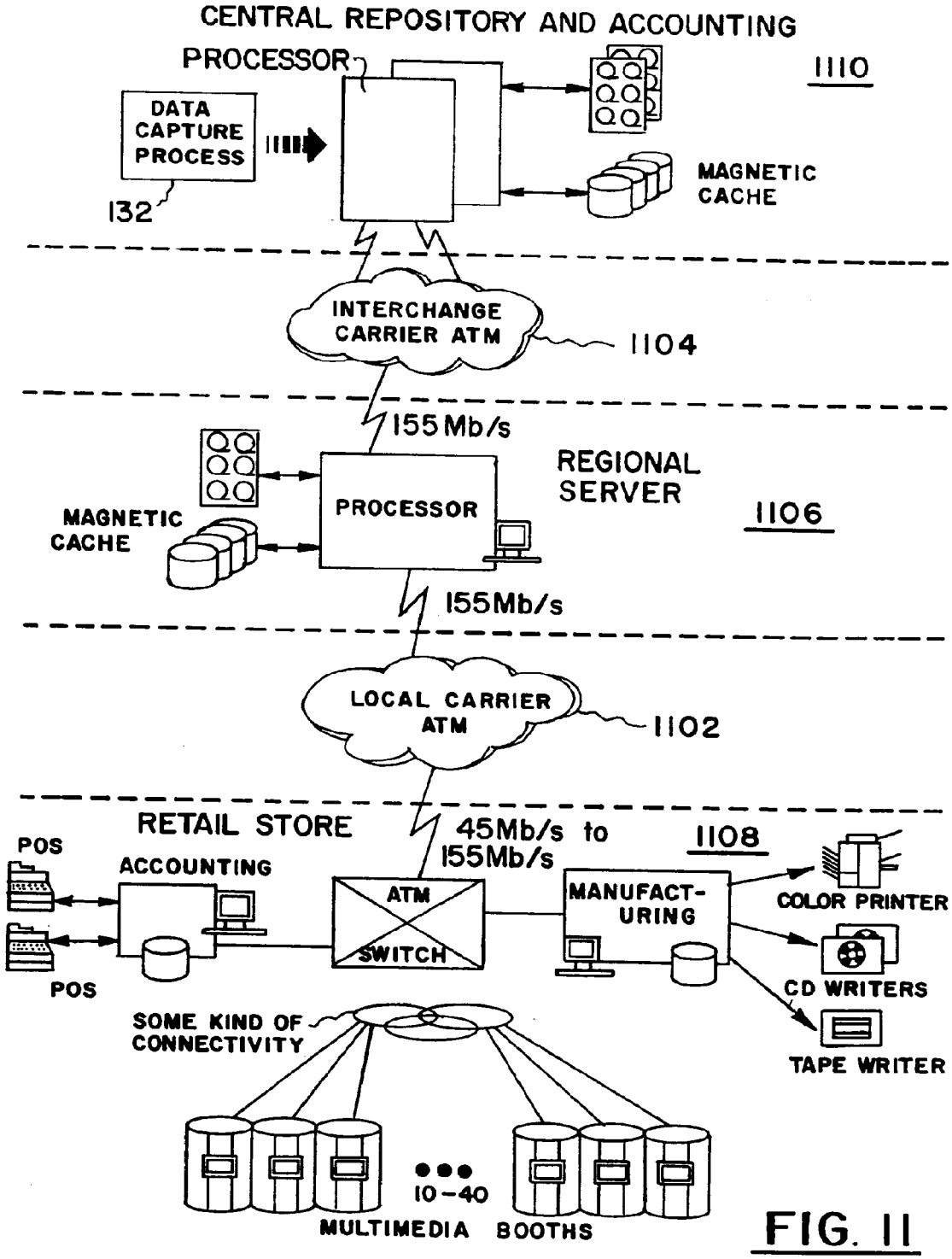

CENTRAL REPOSITORY AND ACCOUNTING
PROCESSOR
1110
DATA CAPTURE PROCESS
132
MAGNETIC CACHE
INTERCHANGE CARRIER ATM
1104
155 Mb/s
REGIONAL SERVER
PROCESSOR
1106
MAGNETIC CACHE
155Mb/s
LOCAL CARRIER ATM
1102
RETAIL STORE
45Mb/s to 155Mb/s
1108
POS
ACCOUNTING
ATM SWITCH
MANUFACT-URING
COLOR PRINTER
CD WRITERS
TAPE WRITER
POS
SOME KIND OF CONNECTIVITY
10-40
MULTIMEDIA BOOTHS

FIG. 11

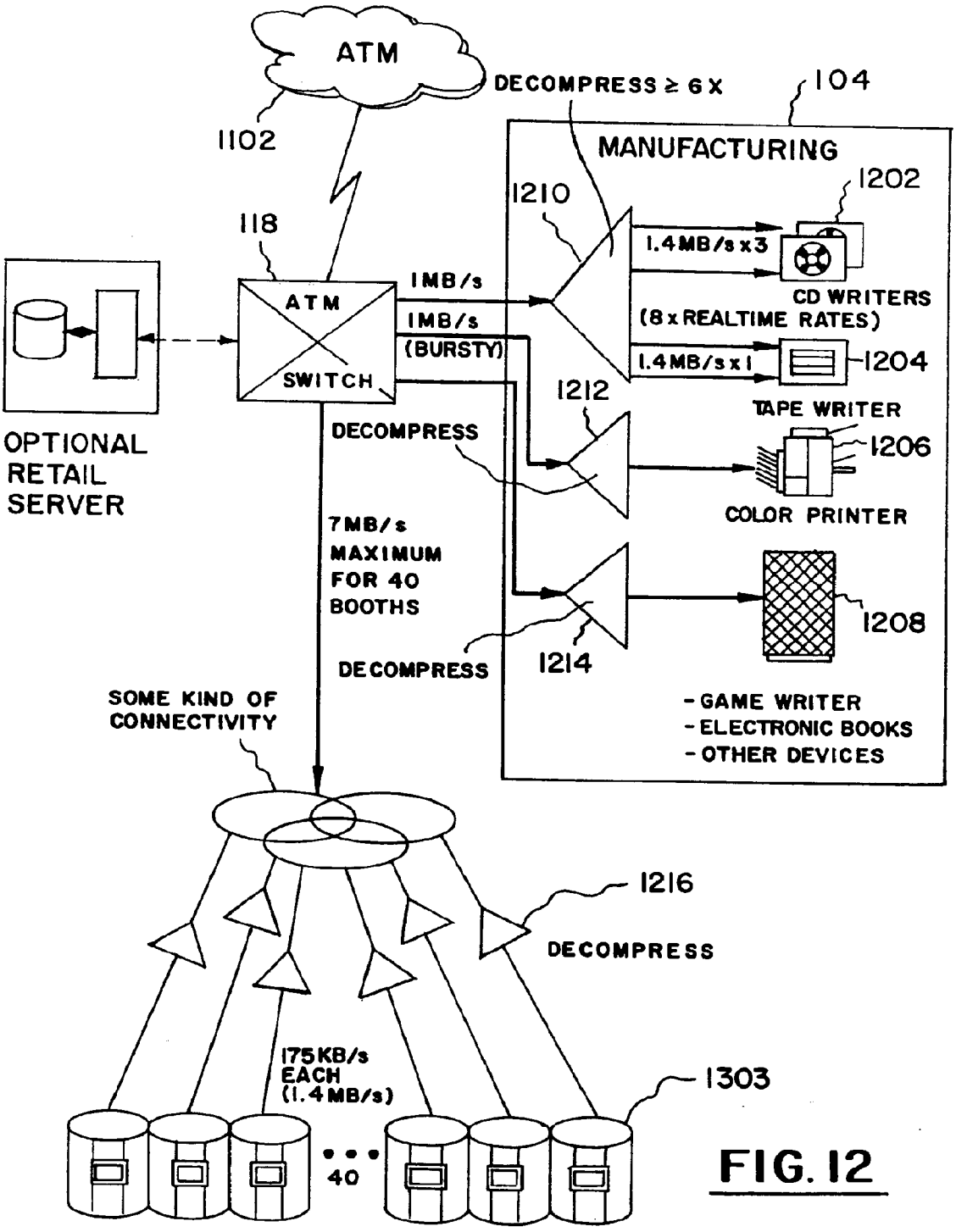

ATM
1102
DECOMPRESS ≥ 6X
104
MANUFACTURING
1202
1210
118
1.4MB/s x3
CD WRITERS
(8 x REALTIME RATES)
1MB/s
ATM
1MB/s
(BURSTY)
1.4MB/s x1
1204
SWITCH
1212
TAPE WRITER
1206
DECOMPRESS
OPTIONAL
RETAIL
SERVER
COLOR PRINTER
7MB/s
MAXIMUM
FOR 40
BOOTHS
1208
DECOMPRESS
1214
- GAME WRITER
- ELECTRONIC BOOKS
- OTHER DEVICES
SOME KIND OF
CONNECTIVITY
1216
DECOMPRESS
175KB/s
EACH
(1.4MB/s)
1303
40

FIG. 12

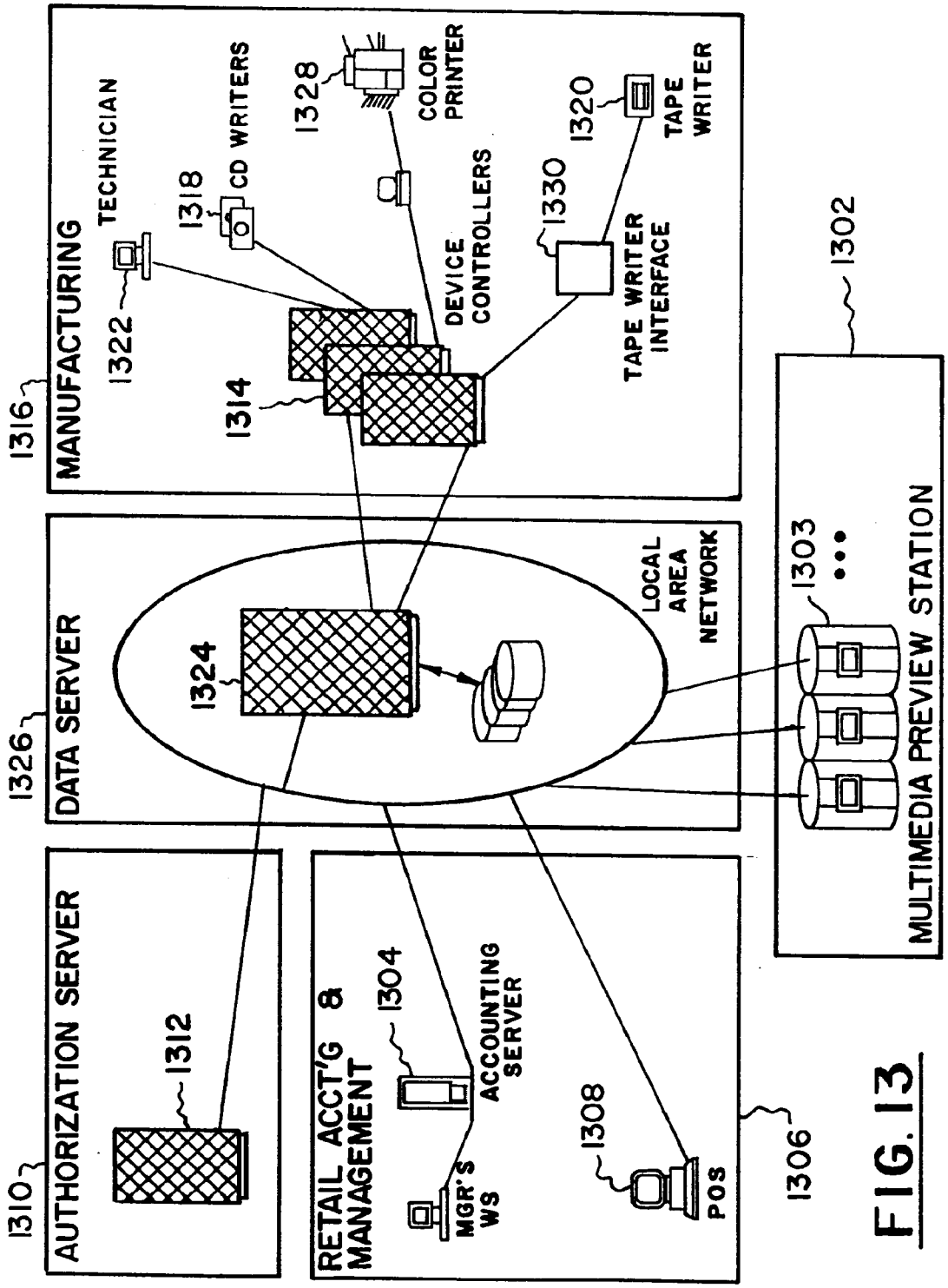

MANUFACTURING
1316
TECHNICIAN
1322
CD WRITERS
1318
COLOR PRINTER
1328
DEVICE CONTROLLERS
1314
TAPE WRITER INTERFACE
1330
TAPE WRITER
1320
DATA SERVER
1326
1324
LOCAL AREA NETWORK
1302
1303
MULTIMEDIA PREVIEW STATION
AUTHORIZATION SERVER
1310
1312
RETAIL ACCT'G & MANAGEMENT
1304
ACCOUNTING SERVER
MGR'S WS
1308
POS
1306

FIG. 13

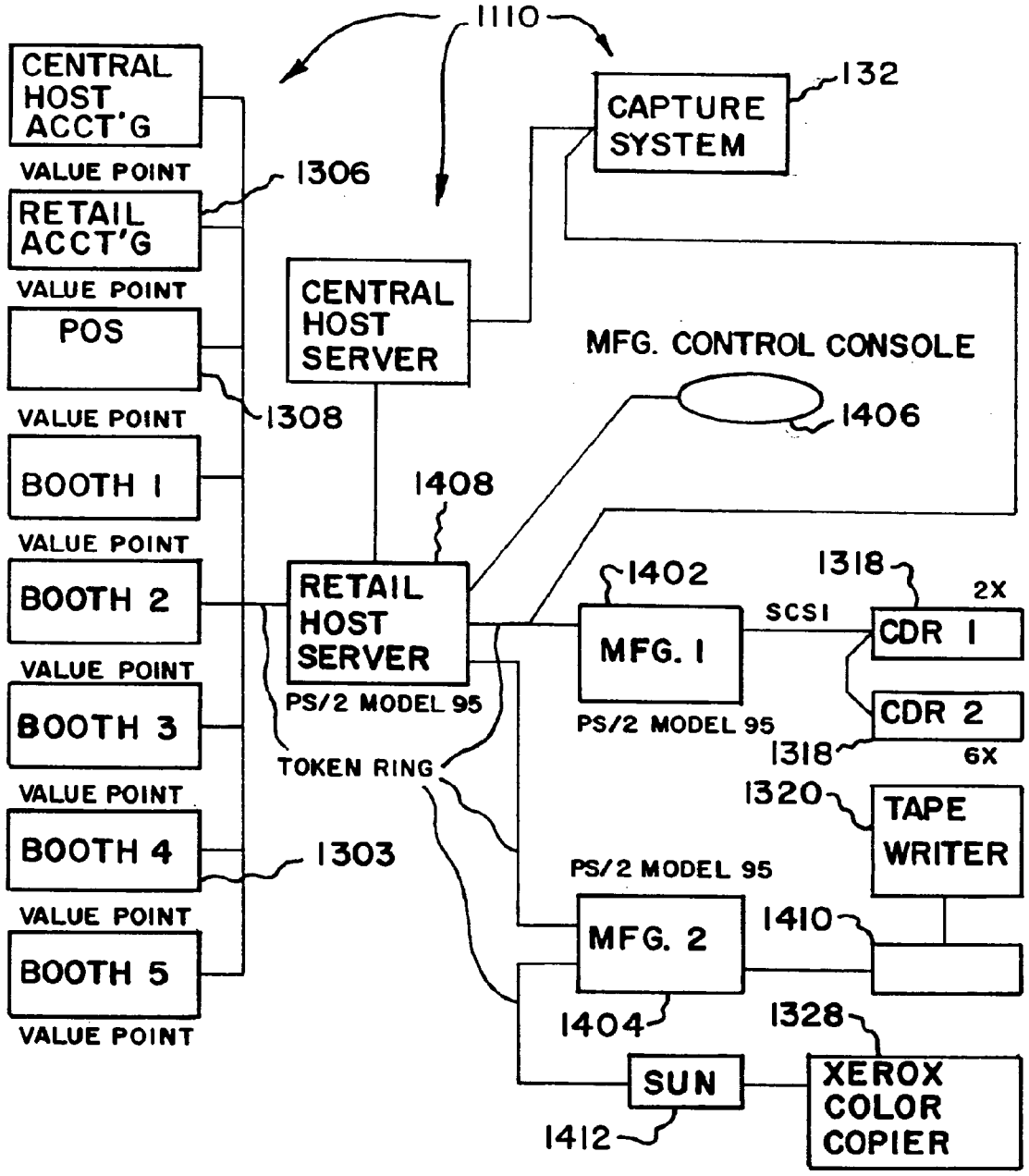

1110
CENTRAL HOST ACCT'G
VALUE POINT
RETAIL ACCT'G
VALUE POINT
POS
VALUE POINT
BOOTH 1
VALUE POINT
BOOTH 2
VALUE POINT
BOOTH 3
VALUE POINT
BOOTH 4
VALUE POINT
BOOTH 5
VALUE POINT
1306
1308
1303
CAPTURE SYSTEM
132
CENTRAL HOST SERVER
MFG. CONTROL CONSOLE
1406
1408
RETAIL HOST SERVER
PS/2 MODEL 95
TOKEN RING
1402
MFG. 1
PS/2 MODEL 95
SCSI
1318
2X
CDR 1
CDR 2
1318
6X
1320
TAPE WRITER
PS/2 MODEL 95
MFG. 2
1410
1404
1328
SUN
XEROX COLOR COPIER
1412

FIG. 14

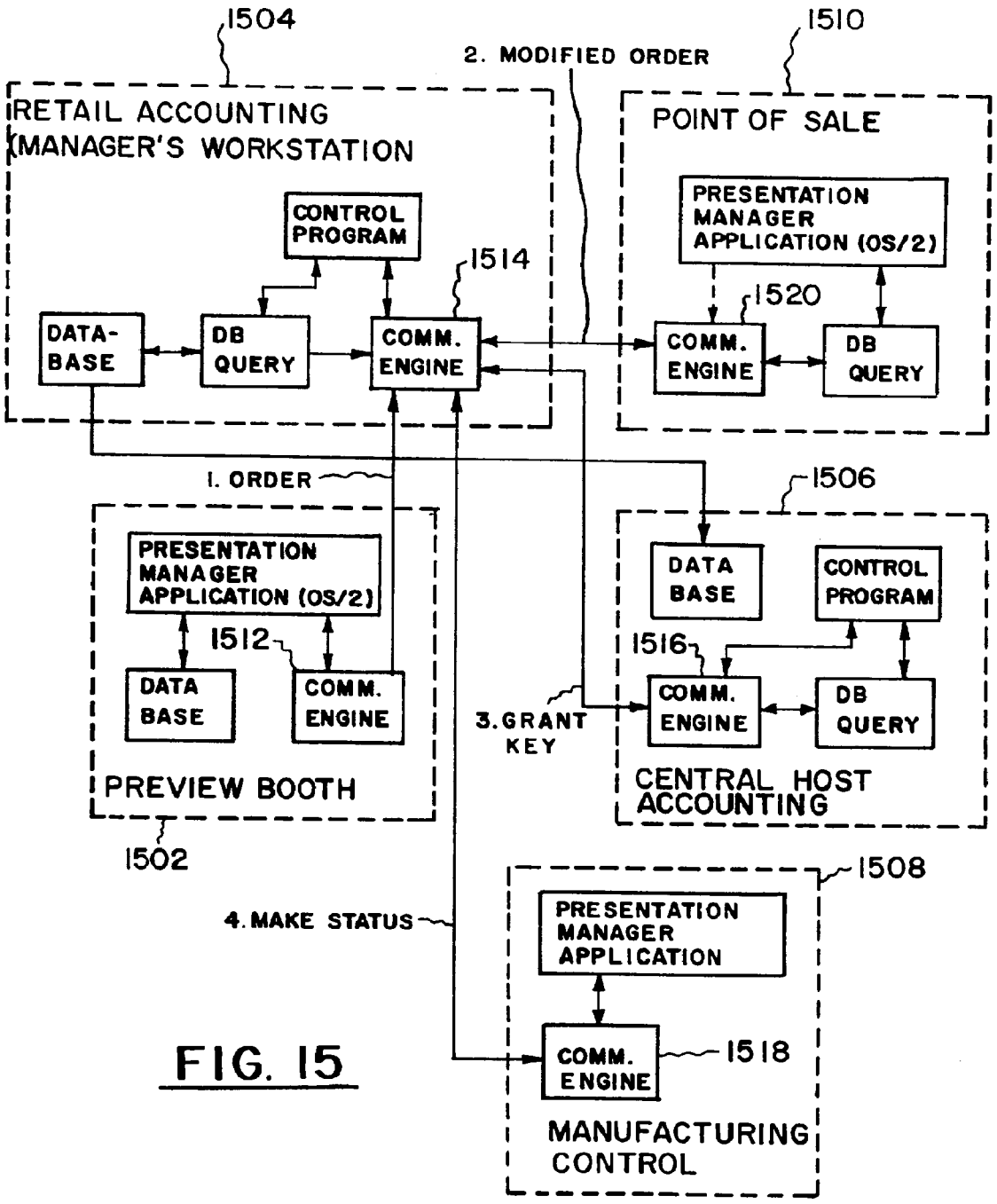

1504
2. MODIFIED ORDER
1510
RETAIL ACCOUNTING
MANAGER'S WORKSTATION
CONTROL PROGRAM
1514
DATA-BASE
DB QUERY
COMM. ENGINE
POINT OF SALE
PRESENTATION MANAGER APPLICATION (OS/2)
1520
COMM. ENGINE
DB QUERY
1. ORDER
1506
PRESENTATION MANAGER APPLICATION (OS/2)
1512
DATA BASE
COMM. ENGINE
PREVIEW BOOTH
1502
DATA BASE
CONTROL PROGRAM
1516
3. GRANT KEY
COMM. ENGINE
DB QUERY
CENTRAL HOST ACCOUNTING
1508
4. MAKE STATUS
PRESENTATION MANAGER APPLICATION
COMM. ENGINE
1518
MANUFACTURING CONTROL

FIG. 15

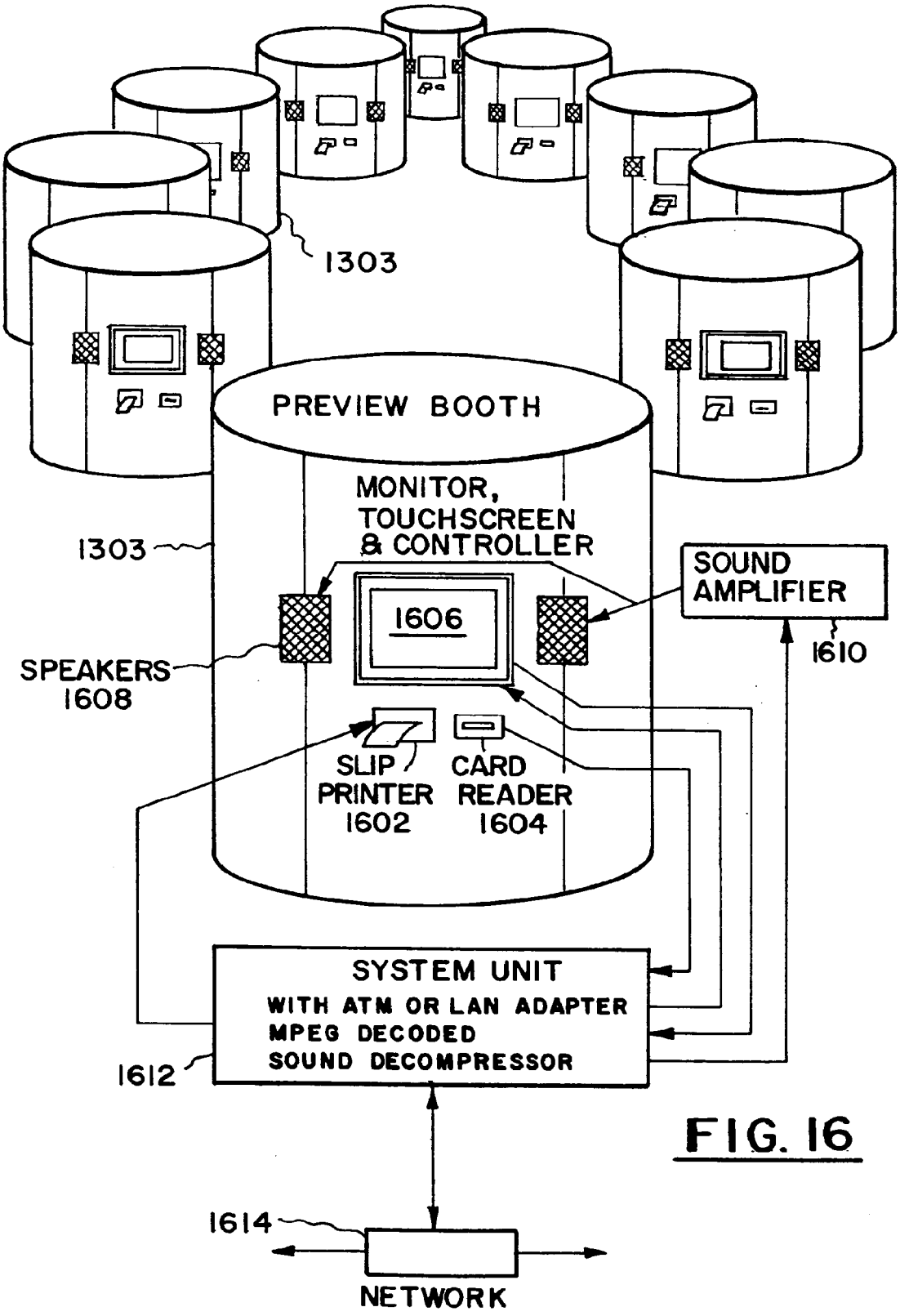

1303
PREVIEW BOOTH
MONITOR,
TOUCHSCREEN
& CONTROLLER
1303
SOUND
AMPLIFIER
1606
1610
SPEAKERS
1608
SLIP
PRINTER
1602
CARD
READER
1604
SYSTEM UNIT
WITH ATM OR LAN ADAPTER
MPEG DECODED
SOUND DECOMPRESSOR
1612
1614
NETWORK

FIG. 16

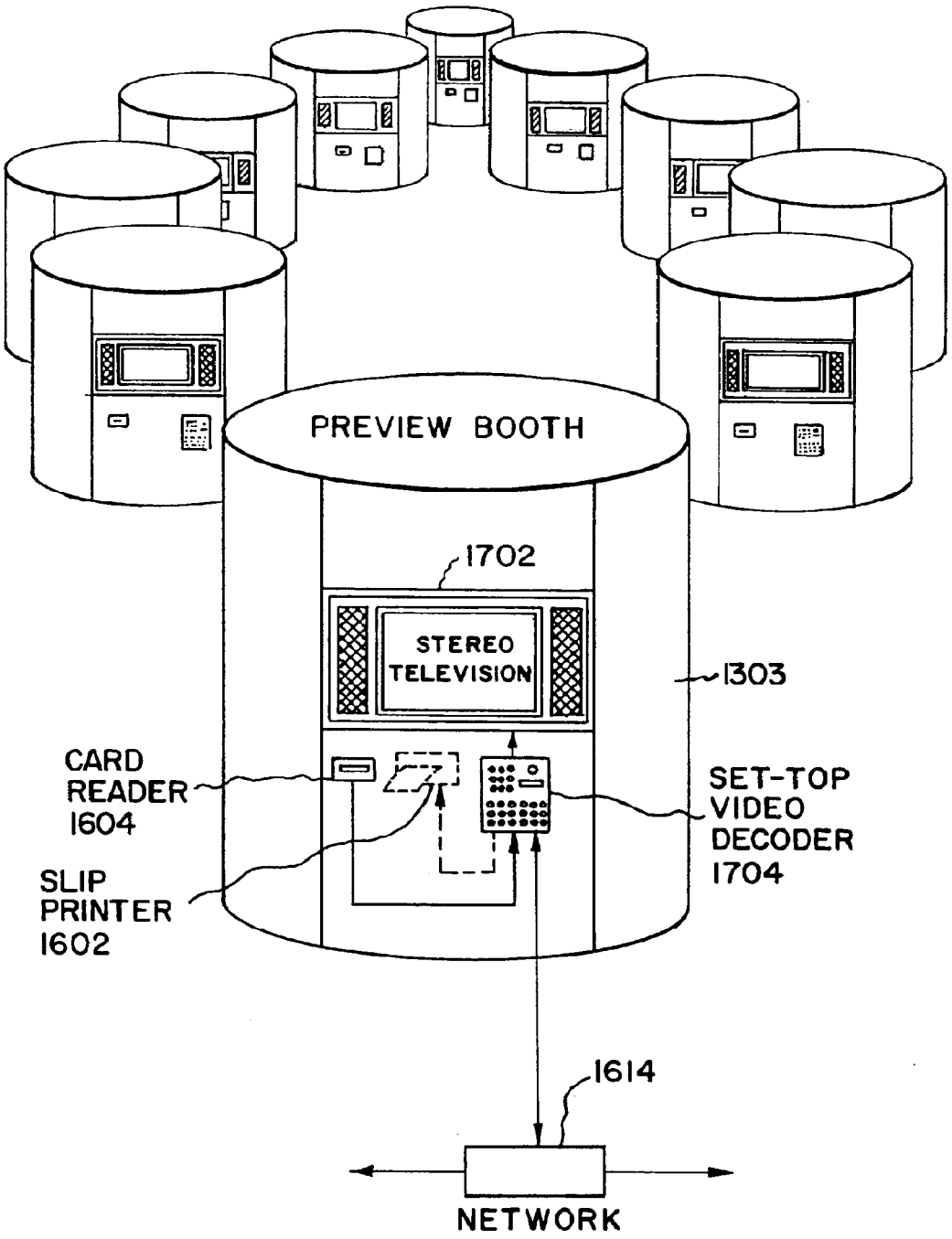

PREVIEW BOOTH
1702
STEREO TELEVISION
1303
CARD READER 1604
SLIP PRINTER 1602
SET-TOP VIDEO DECODER 1704
1614
NETWORK

FIG. 17

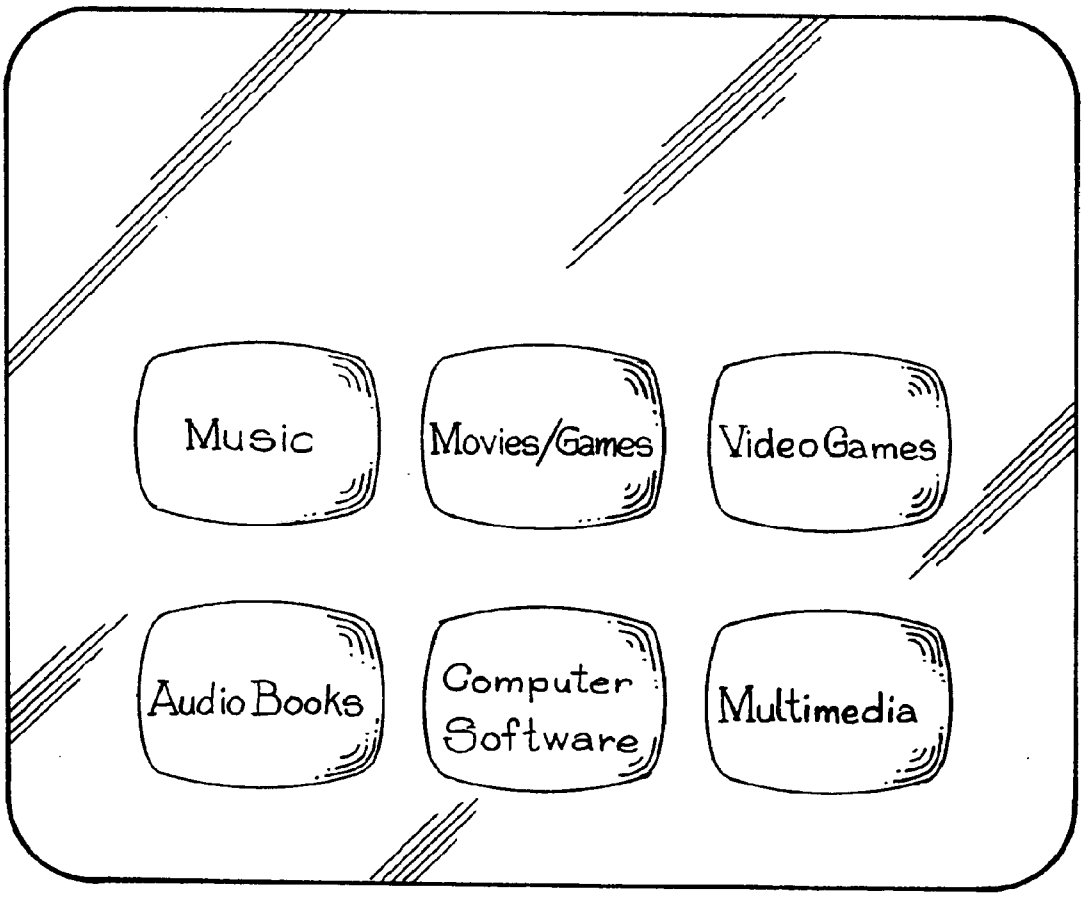

Music
Movies/Games
Video Games
Audio Books
Computer Software
Multimedia

FIG. 18

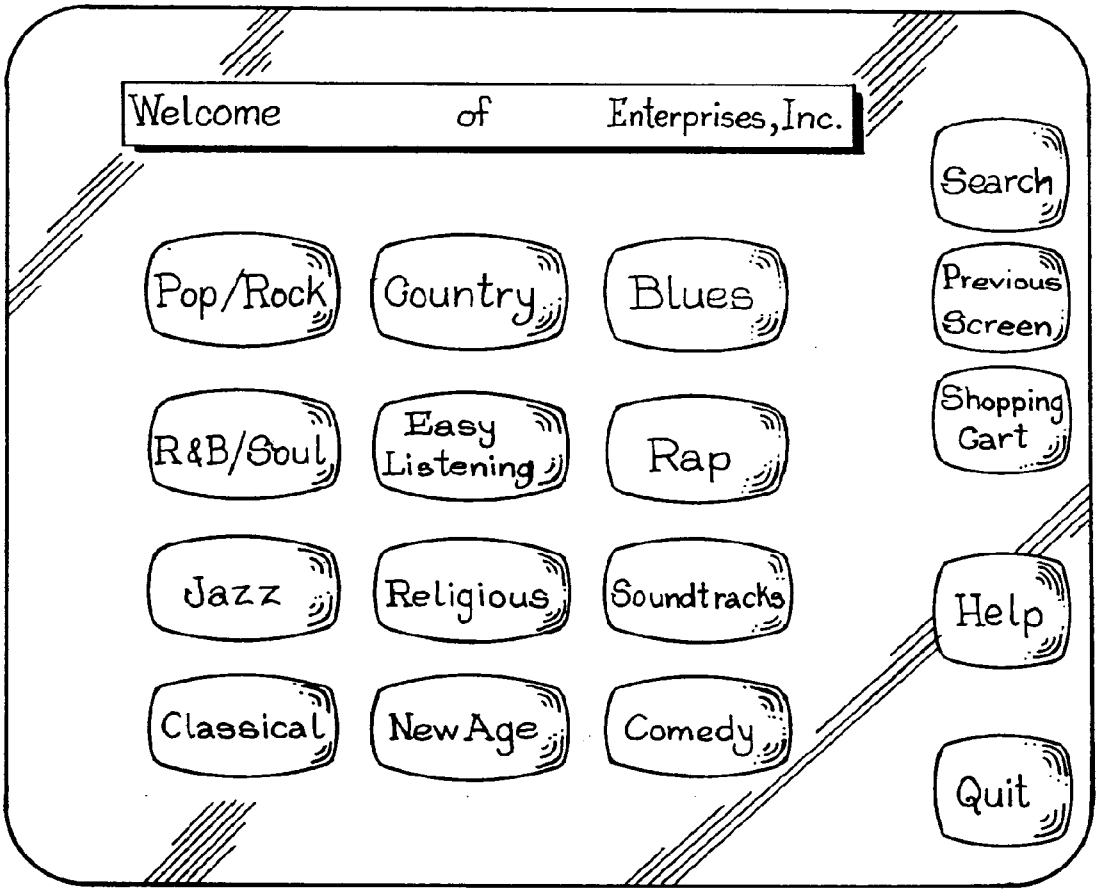

Welcome of Enterprises, Inc.
Search
Pop/Rock
Country
Blues
Previous Screen
R&B/Soul
Easy Listening
Rap
Shopping Cart
Jazz
Religious
Soundtracks
Help
Classical
New Age
Comedy
Quit

FIG. 19

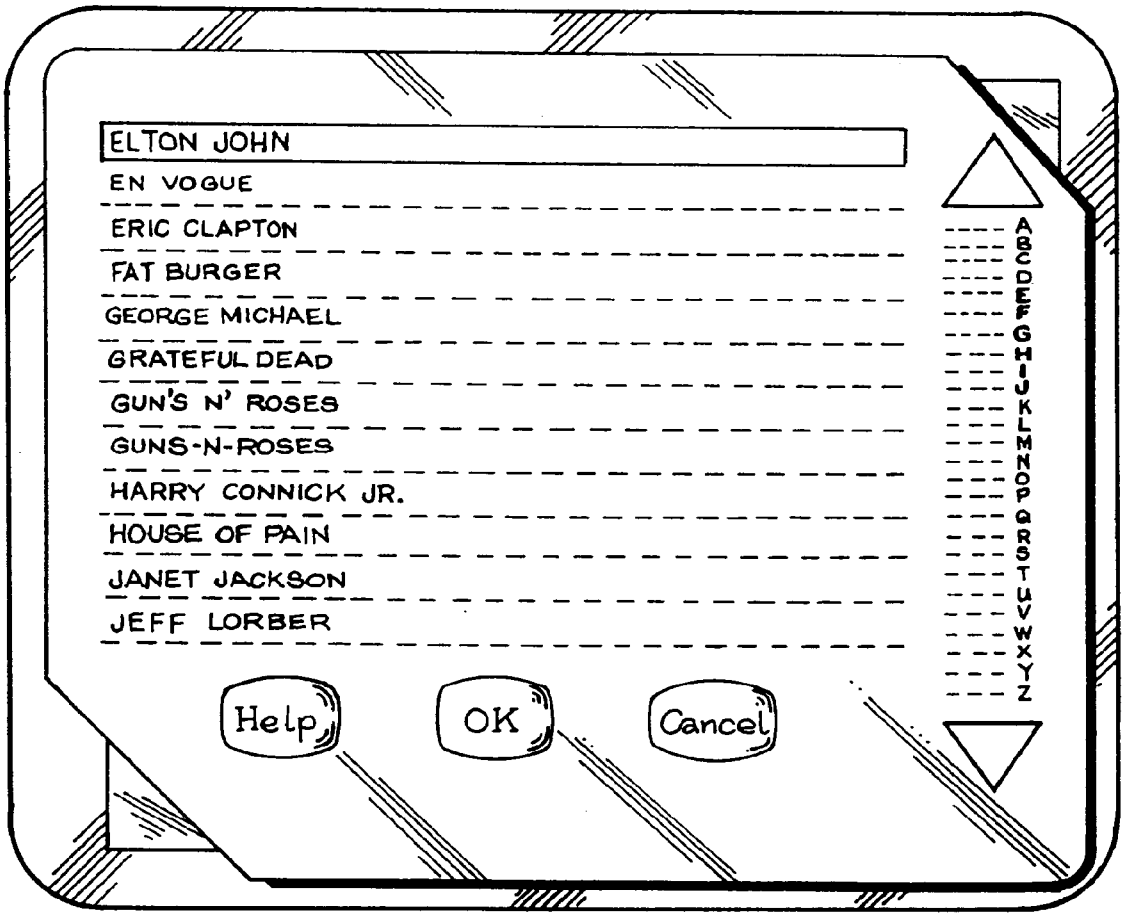

ELTON JOHN
EN VOGUE
ERIC CLAPTON
FAT BURGER
GEORGE MICHAEL
GRATEFUL DEAD
GUN'S N' ROSES
GUNS-N-ROSES
HARRY CONNICK JR.
HOUSE OF PAIN
JANET JACKSON
JEFF LORBER
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Help
OK
Cancel

FIG. 20

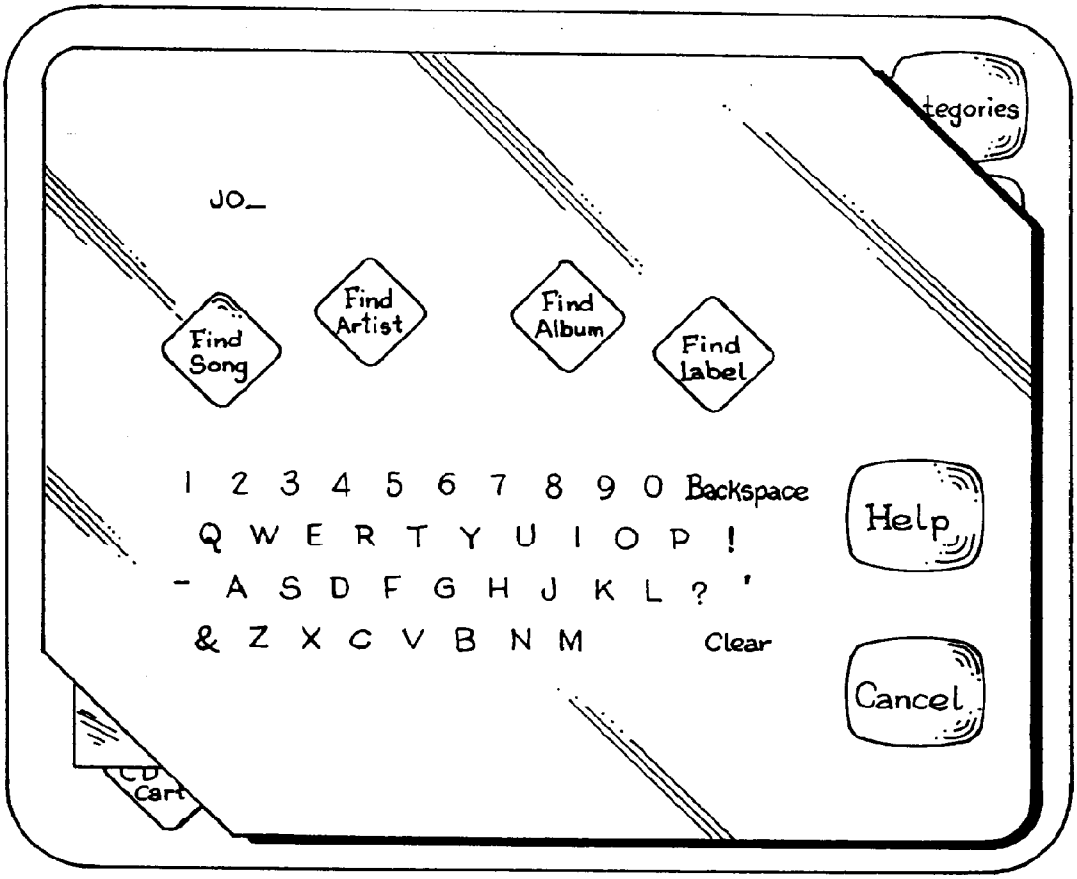

tegories
JO_
Find Song
Find Artist
Find Album
Find label
1 2 3 4 5 6 7 8 9 0 Backspace
Q W E R T Y U I O P !
- A S D F G H J K L ? '
& Z X C V B N M
Clear
Help
Cancel
CD Cart

FIG. 21

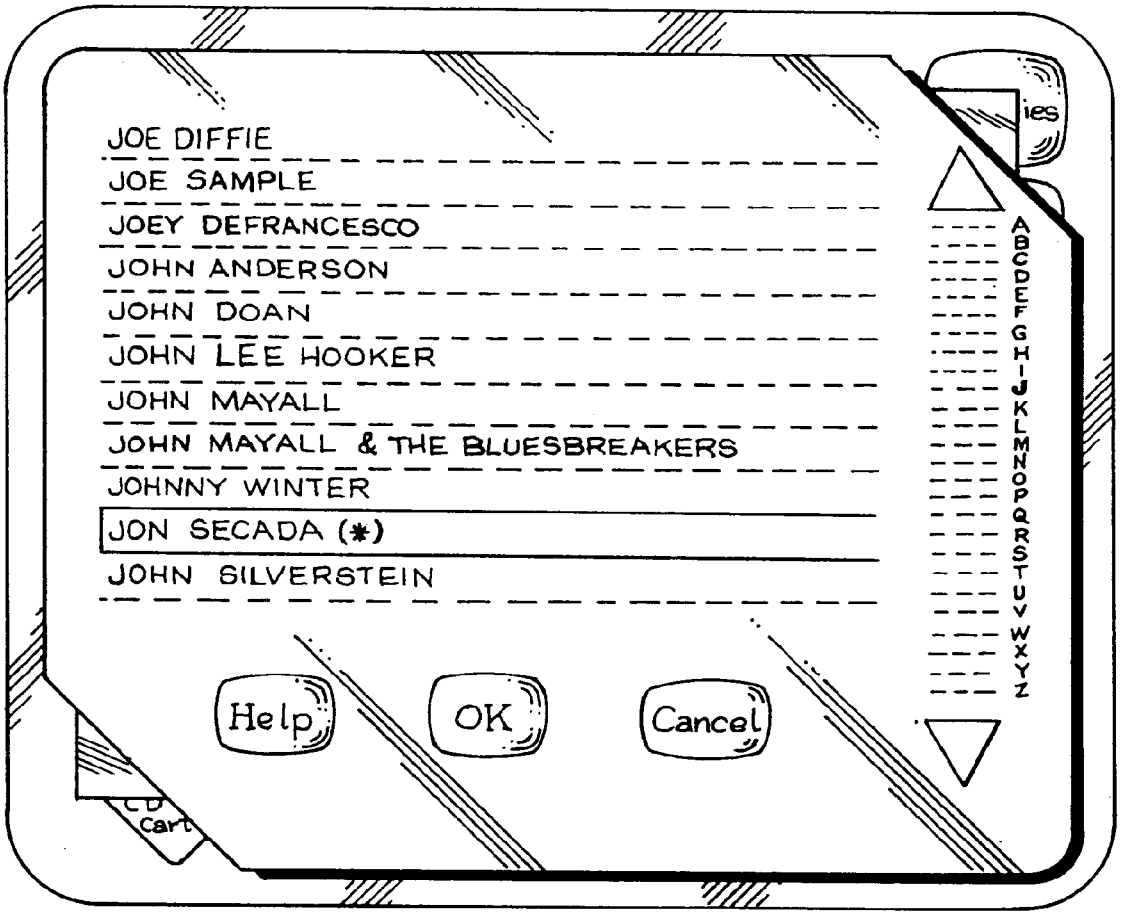

JOE DIFFIE
JOE SAMPLE
JOEY DEFRANCESCO
JOHN ANDERSON
JOHN DOAN
JOHN LEE HOOKER
JOHN MAYALL
JOHN MAYALL & THE BLUESBREAKERS
JOHNNY WINTER
JON SECADA (*)
JOHN SILVERSTEIN
Help
OK
Cancel
ies
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

FIG. 22

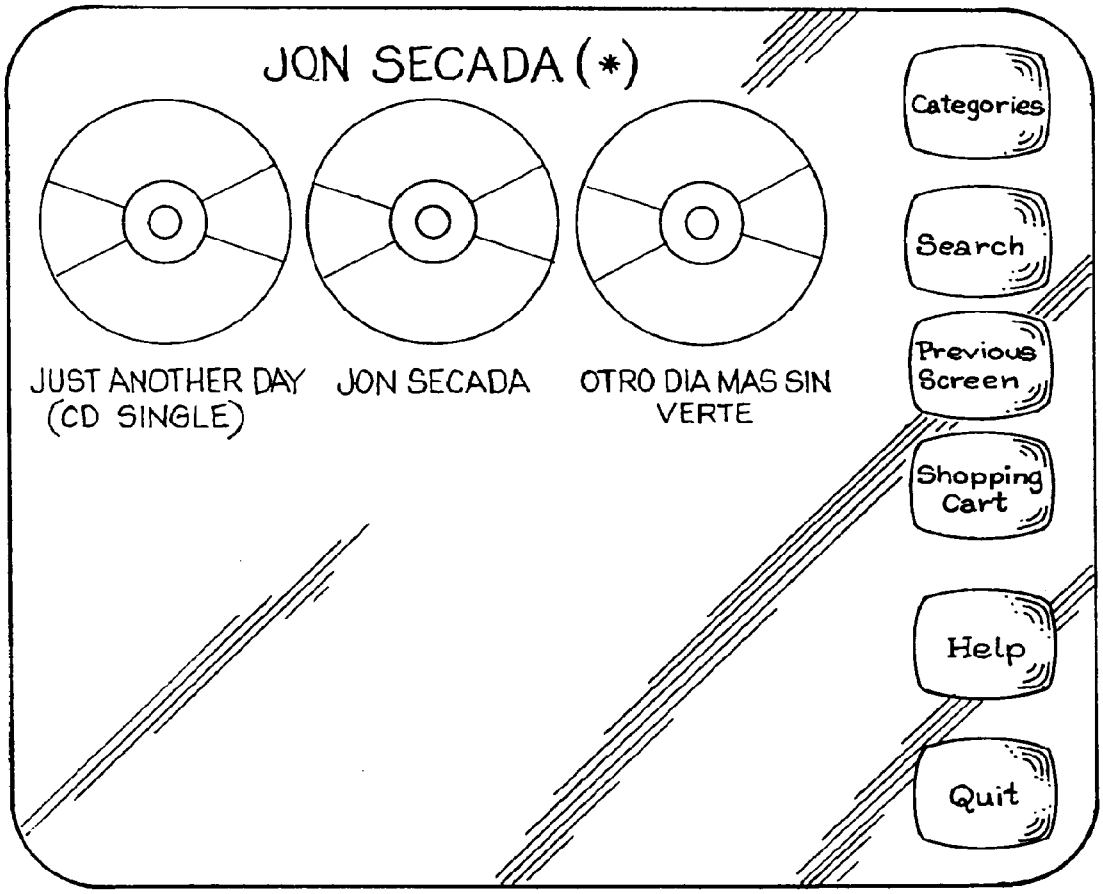

JON SECADA (*)
JUST ANOTHER DAY (CD SINGLE)
JON SECADA
OTRO DIA MAS SIN VERTE
Categories
Search
Previous Screen
Shopping Cart
Help
Quit

FIG. 23

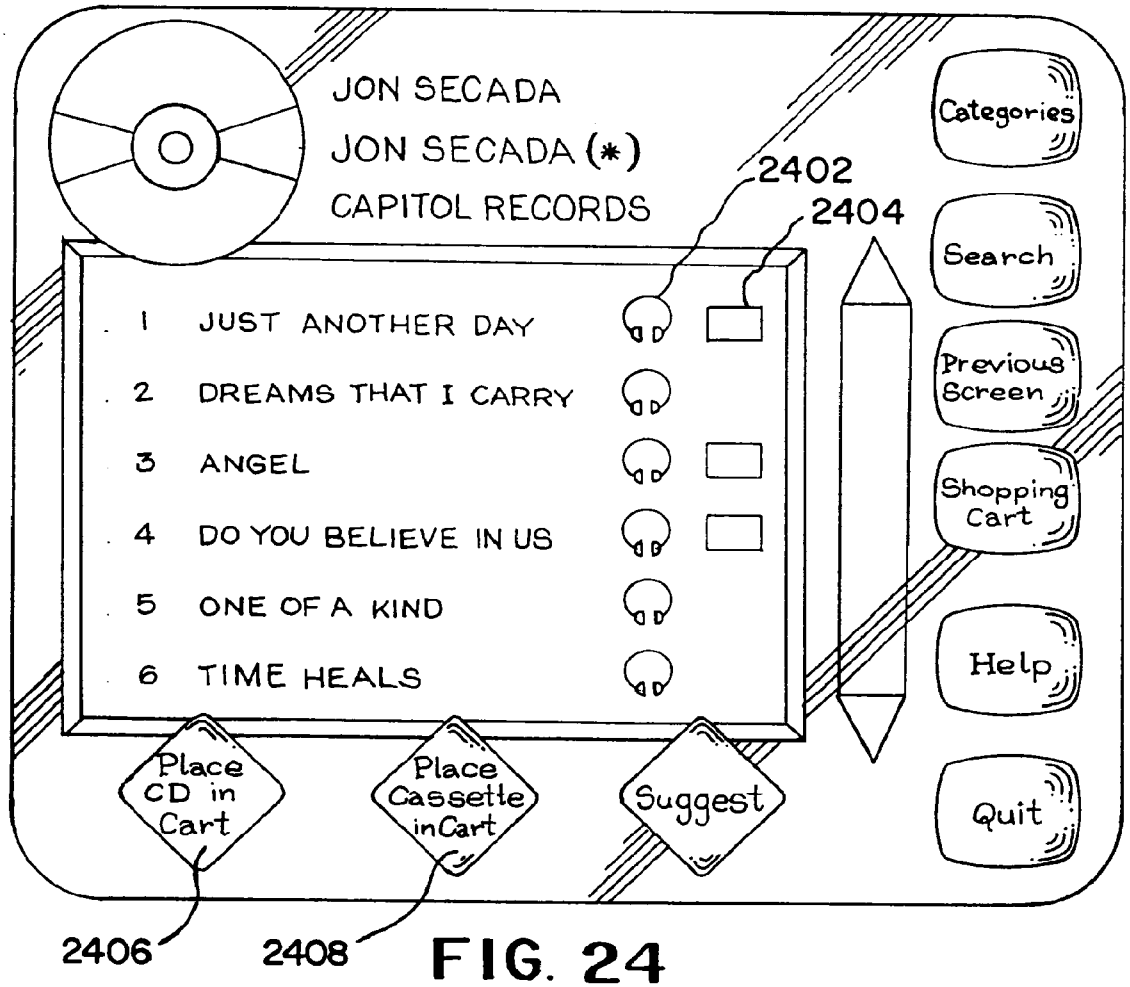

JON SECADA
JON SECADA (*)
CAPITOL RECORDS
2402
2404
1 JUST ANOTHER DAY
2 DREAMS THAT I CARRY
3 ANGEL
4 DO YOU BELIEVE IN US
5 ONE OF A KIND
6 TIME HEALS
Categories
Search
Previous Screen
Shopping Cart
Help
Quit
Place CD in Cart
Place Cassette in Cart
Suggest
2406
2408

FIG. 24

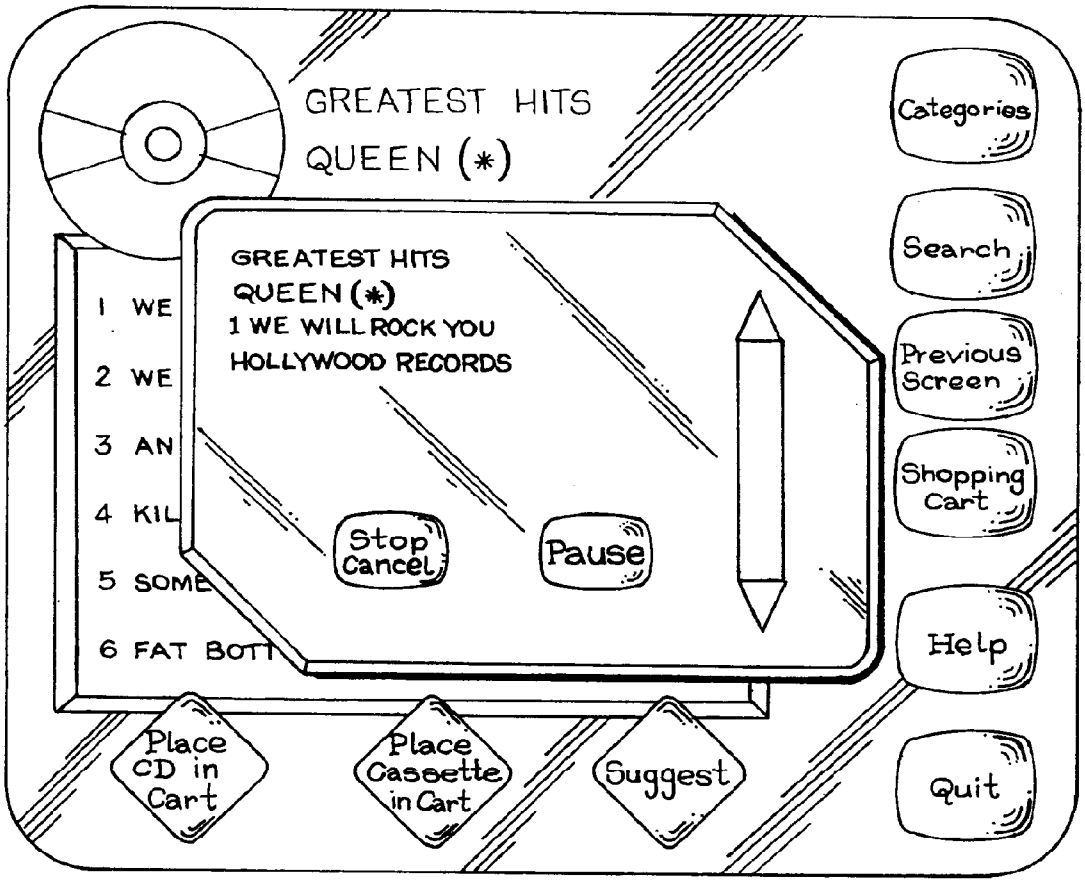

GREATEST HITS
QUEEN (*)
1 WE
2 WE
3 AN
4 KIL
5 SOME
6 FAT BOTT
GREATEST HITS
QUEEN (*)
1 WE WILL ROCK YOU
HOLLYWOOD RECORDS
Stop Cancel
Pause
Categories
Search
Previous Screen
Shopping Cart
Help
Quit
Place CD in Cart
Place Cassette in Cart
Suggest

FIG. 25

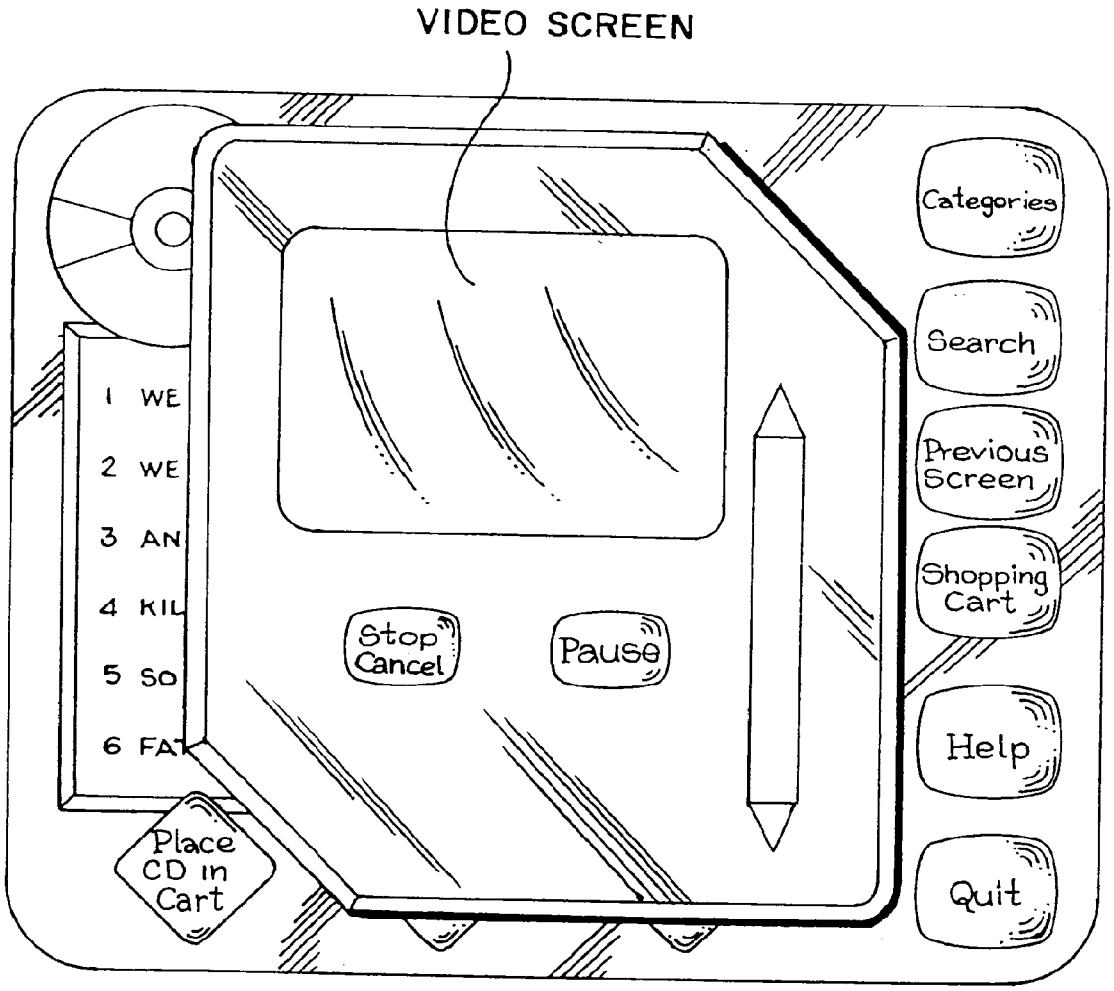

VIDEO SCREEN
Categories
Search
Previous Screen
Shopping Cart
Help
Quit
1 WE
2 WE
3 AN
4 KIL
5 SO
6 FA
Stop Cancel
Pause
Place CD in Cart

FIG. 26

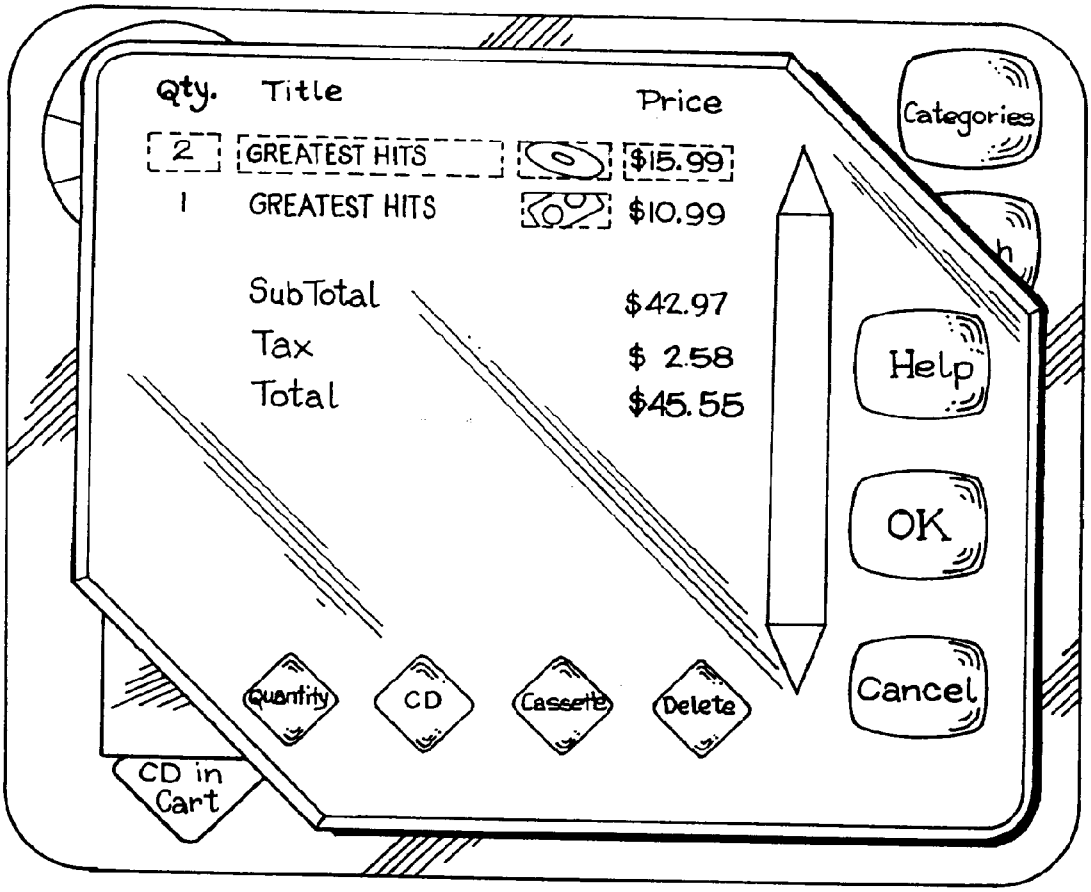

Qty.
Title
Price
2
GREATEST HITS
$15.99
1
GREATEST HITS
$10.99
SubTotal
$42.97
Tax
$ 2.58
Total
$45.55
Categories
Help
OK
Cancel
Quantity
CD
Cassette
Delete
CD in Cart

FIG. 27

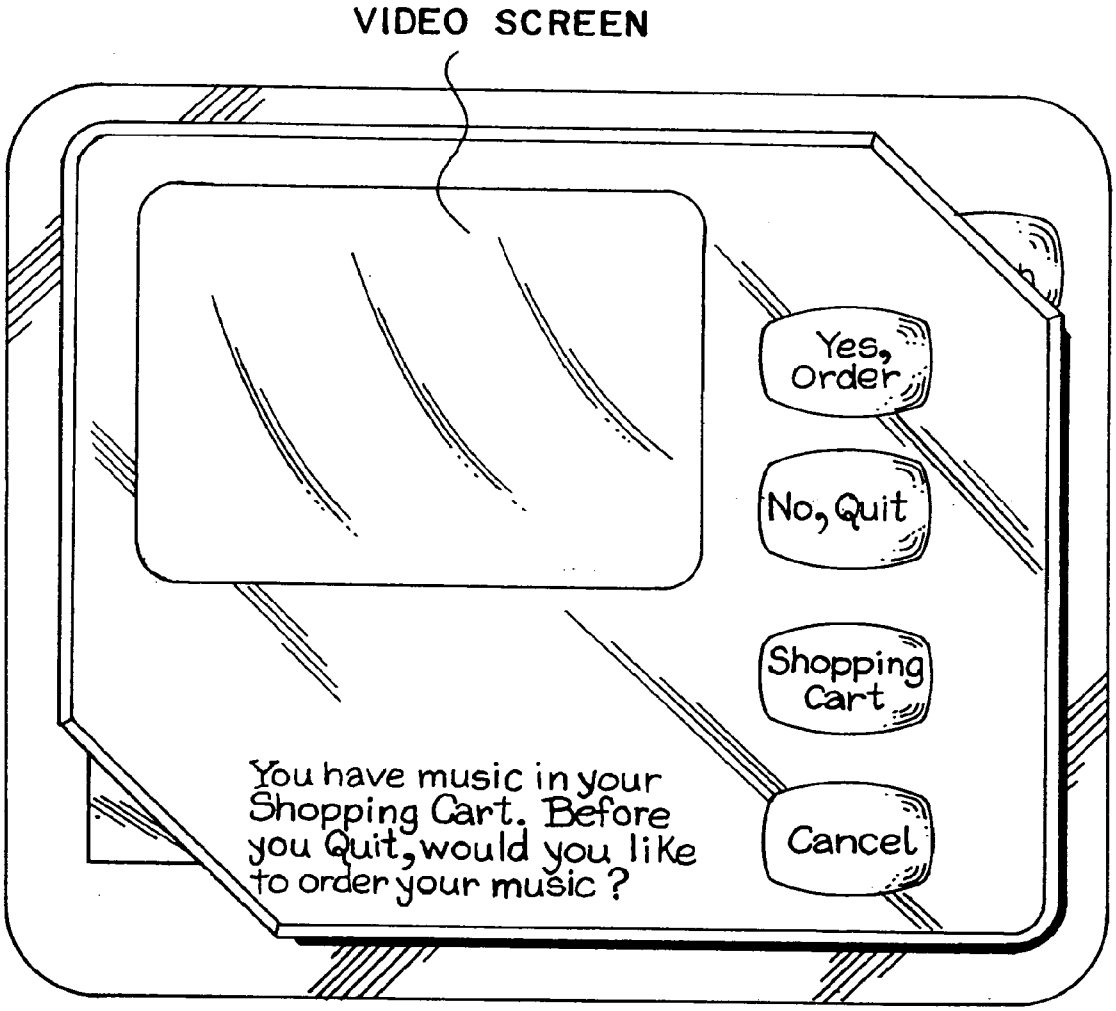

VIDEO SCREEN
Yes, Order
No, Quit
Shopping Cart
Cancel
You have music in your Shopping Cart. Before you Quit, would you like to order your music?

FIG. 28

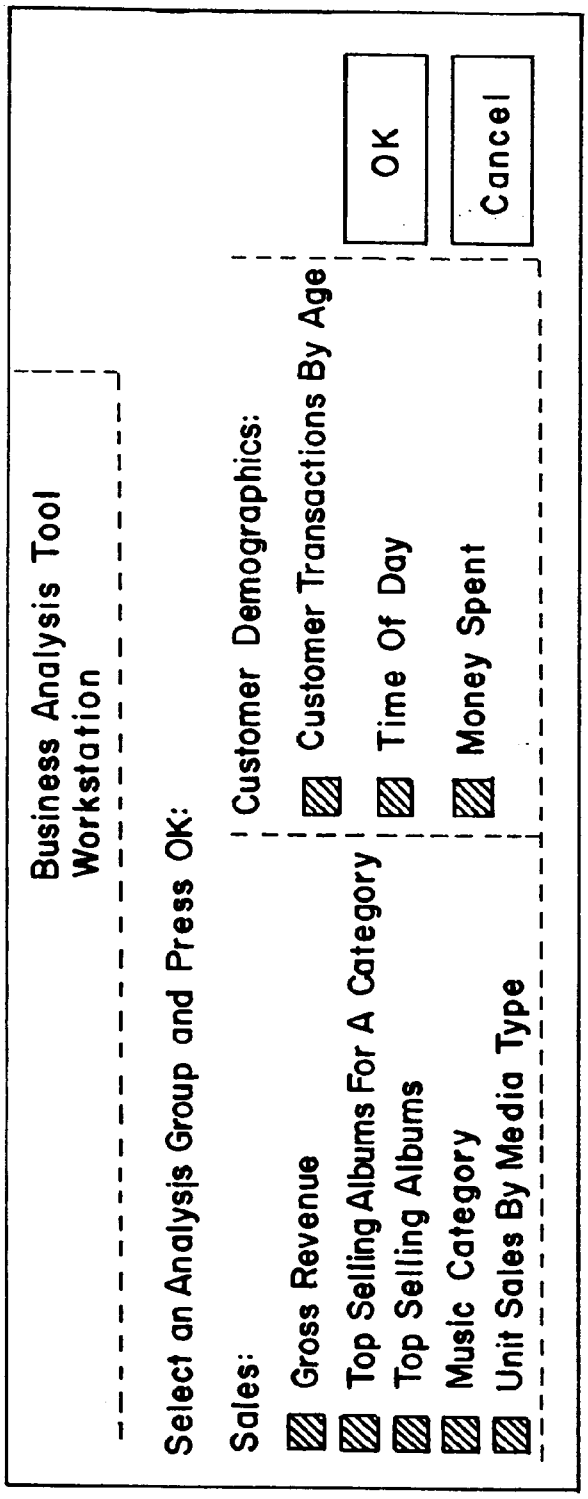

Business Analysis Tool
Workstation
Select an Analysis Group and Press OK:
Sales:
Gross Revenue
Top Selling Albums For A Category
Top Selling Albums
Music Category
Unit Sales By Media Type
Customer Demographics:
Customer Transactions By Age
Time Of Day
Money Spent
OK
Cancel

FIG. 29

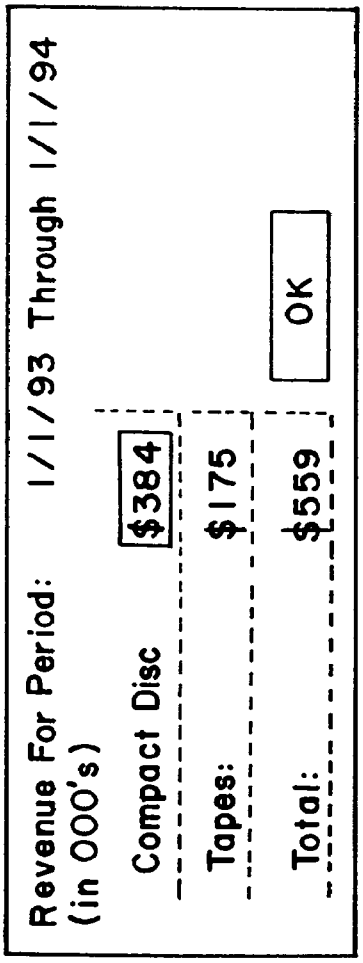

Revenue For Period: 1/1/93 Through 1/1/94
(in 000's)
Compact Disc
$384
Tapes:
$175
Total:
$559
OK

FIG. 30

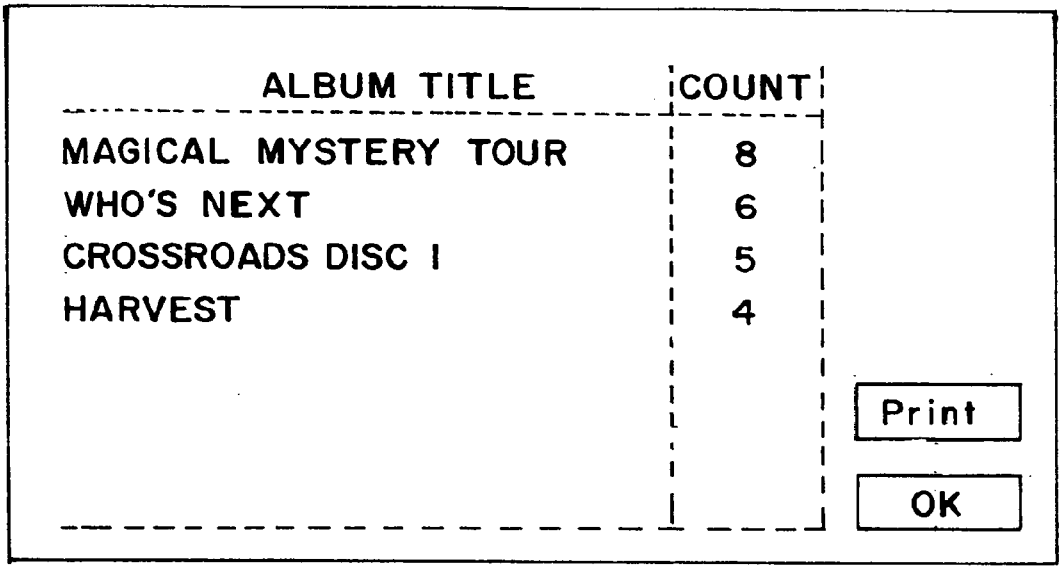

ALBUM TITLE
COUNT
MAGICAL MYSTERY TOUR
8
WHO'S NEXT
6
CROSSROADS DISC I
5
HARVEST
4
Print
OK

FIG. 31

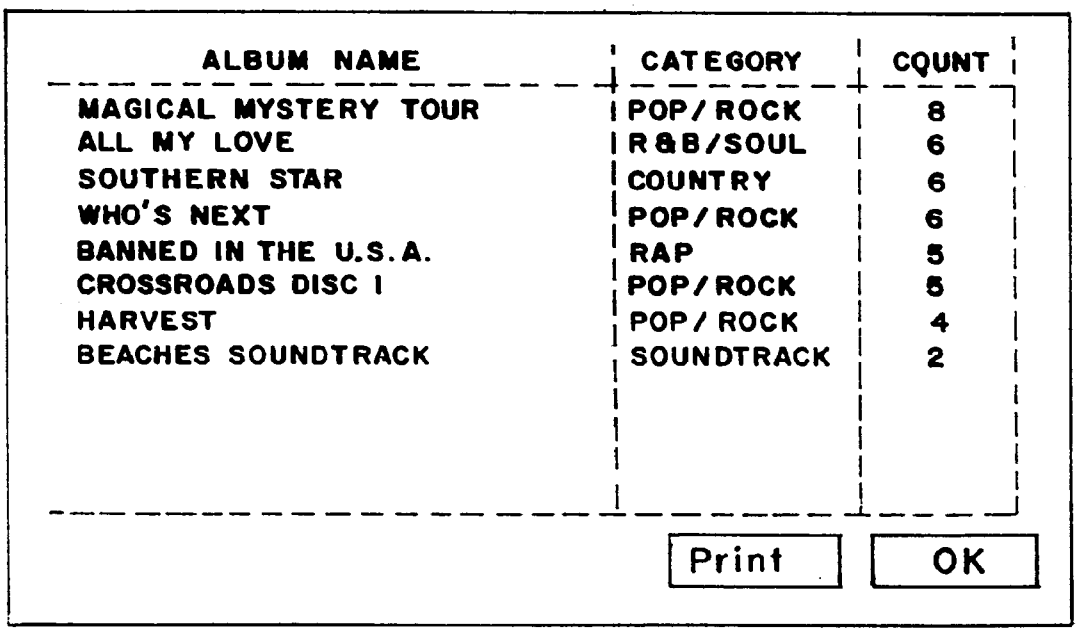

ALBUM NAME
CATEGORY
COUNT
MAGICAL MYSTERY TOUR
POP/ROCK
8
ALL MY LOVE
R&B/SOUL
6
SOUTHERN STAR
COUNTRY
6
WHO'S NEXT
POP/ROCK
6
BANNED IN THE U.S.A.
RAP
5
CROSSROADS DISC I
POP/ROCK
5
HARVEST
POP/ROCK
4
BEACHES SOUNDTRACK
SOUNDTRACK
2
Print
OK

FIG. 32

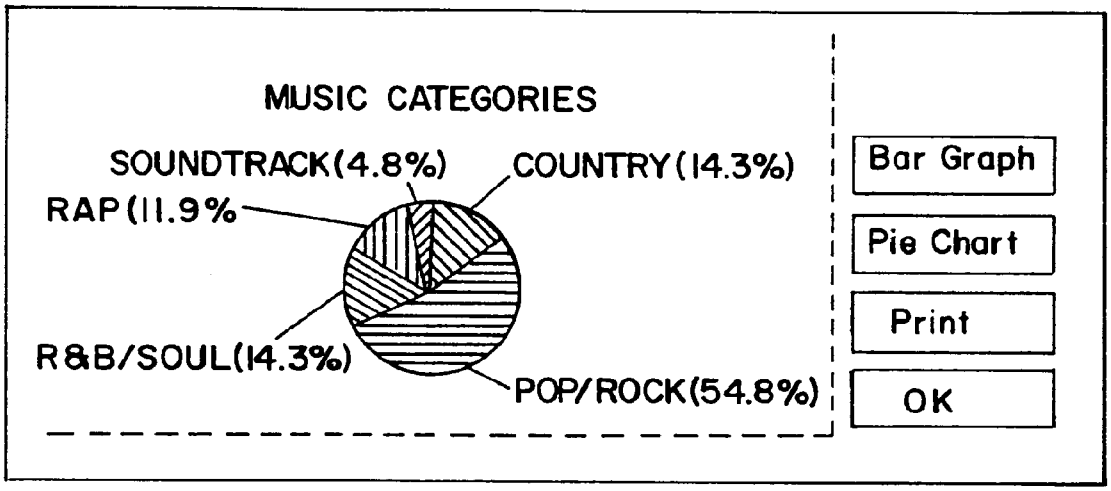

MUSIC CATEGORIES
SOUNDTRACK(4.8%)
COUNTRY(14.3%)
RAP(11.9%
R&B/SOUL(14.3%)
POP/ROCK(54.8%)
Bar Graph
Pie Chart
Print
OK

FIG. 33

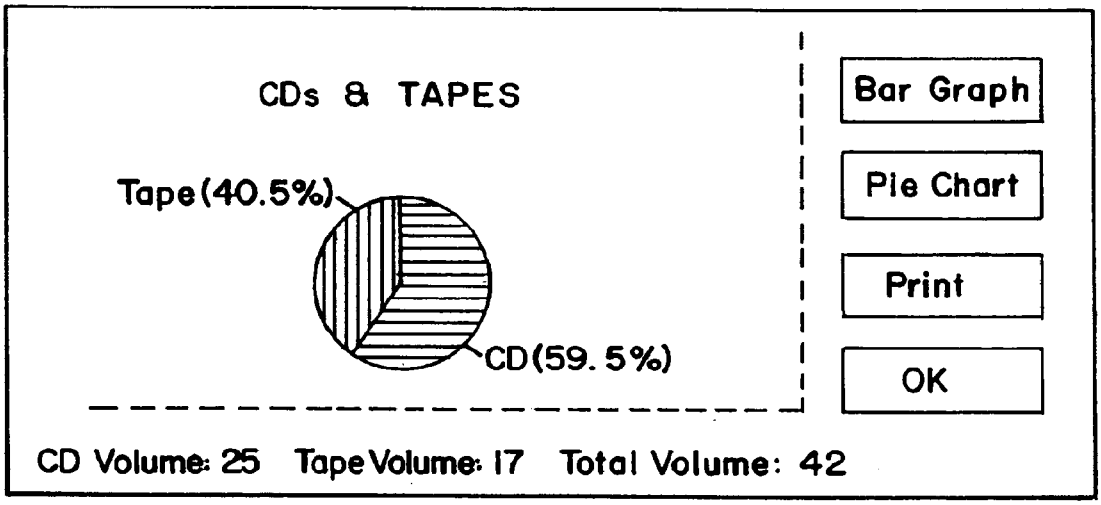

CDs & TAPES
Tape(40.5%)
CD(59.5%)
Bar Graph
Pie Chart
Print
OK
CD Volume: 25 Tape Volume: 17 Total Volume: 42

FIG. 34

Age Groups	CD Volume	Tape Volume	Avg. Units/Transaction
15-19....	9	5	
20-24....			
25-34....			
35-44....			
>45.....			

OK

FIG. 35

Time of Day	Number of Transactions
10:00AM-2:00PM	6
2:00PM-6:00PM	7
6:00PM-Midnight	4

OK

FIG. 36

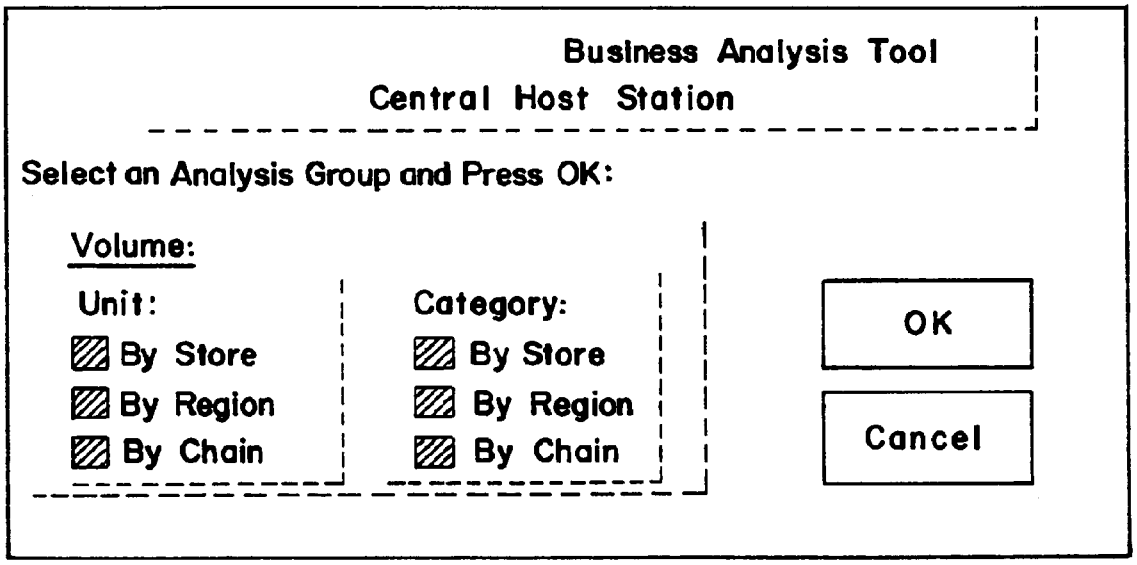

Business Analysis Tool
Central Host Station
Select an Analysis Group and Press OK:
Volume:
Unit:
By Store
By Region
By Chain
Category:
By Store
By Region
By Chain
OK
Cancel

FIG. 37

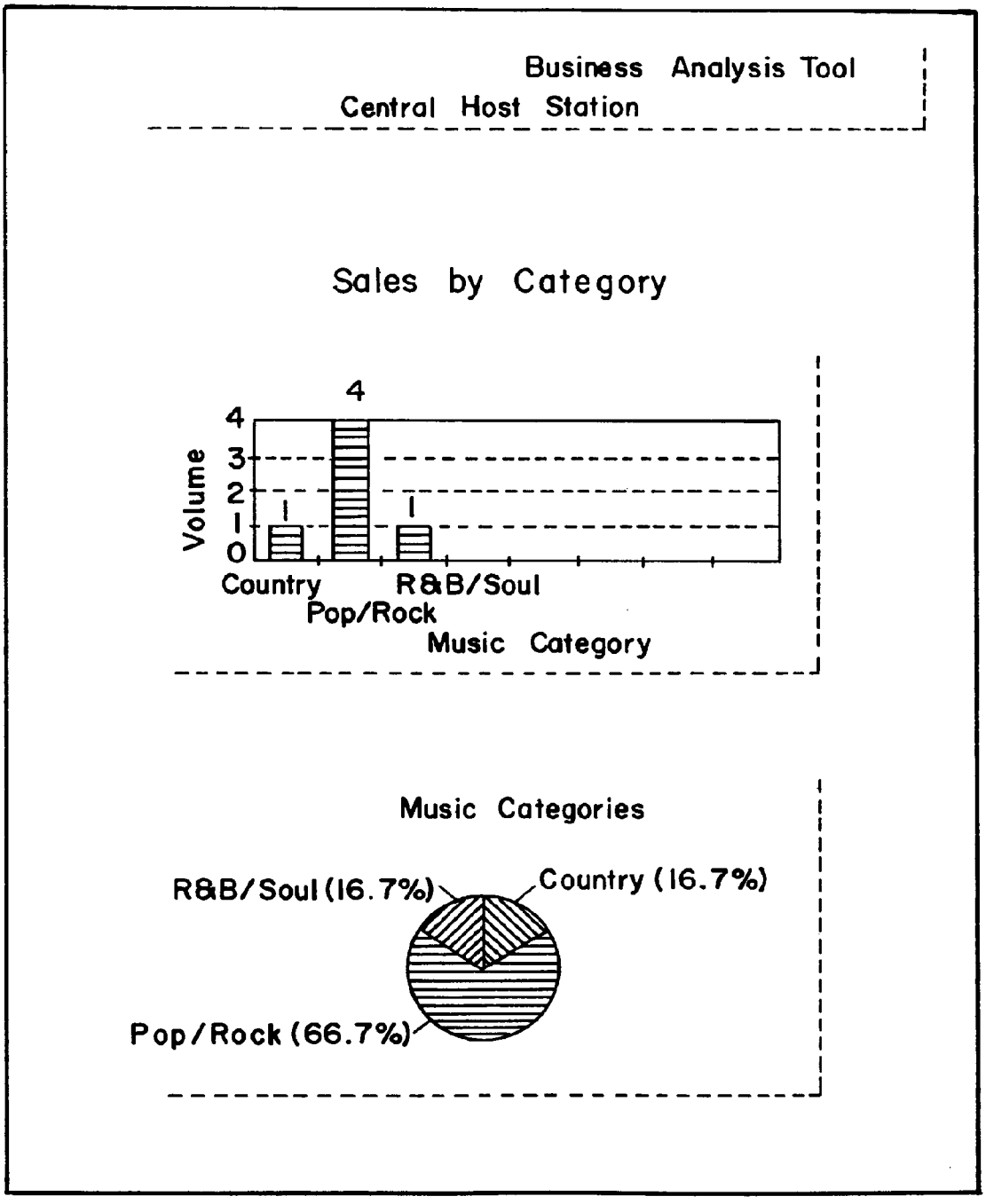

Business Analysis Tool
Central Host Station
Sales by Category
4
4
3
2
1
0
Volume
Country
Pop/Rock
R&B/Soul
Music Category
Music Categories
R&B/Soul (16.7%)
Country (16.7%)
Pop/Rock (66.7%)

FIG. 38

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 651 554 A1**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94420293.6**

(51) Int. Cl.[6] : **H04N 1/21**, G06F 1/00

(22) Date of filing : **25.10.94**

(30) Priority : **29.10.93 US 146371**

(43) Date of publication of application :
03.05.95 Bulletin 95/18

(84) Designated Contracting States :
DE GB

(71) Applicant : **EASTMAN KODAK COMPANY**
343 State Street
Rochester, New York 14650-2201 (US)

(72) Inventor : **Rabbani, Majid, Eastman Kodak Company**
Patent Legal Staff,
343 State Street
Rochester, New York 14650-2201 (US)

Inventor : **Melnychuck, Paul W., Eastman Kodak Company**
Patent Legal Staff,
343 State Street
Rochester, New York 14650-2201 (US)

Inventor : **Axman, Michael Stuart, Eastman Kodak Company**
Patent Legal Staff,
343 State Street
Rochester, New York 14650-2201 (US)

Inventor : **Baradar, Ali R., Eastman Kodak Company**
Patent Legal Staff,
343 State Street
Rochester, New York 14650-2201 (US)

(74) Representative : **Boulard, Denis et al**
Kodak-Pathé
Département Brevets
CRT-Zone Industrielle
F-71102 Chalon-sur-Saône Cédex (FR)

(54) Method and apparatus for the addition and removal of digital watermarks in a hierarchical image storage and retrieval system.

(57) An image processing technique is described in the context of a hierarchical image storage and retrieval system. The method allows for the controlled addition and removal of digital watermarks from selected image components in the hierarchy. The method adds a digital watermark in a selected image resolution component and the means to remove it in an additional image component termed the watermark removal component. The method employs the encryption of the watermark removal component, and decryption with a special key, or password during authorized retrieval. This technique allows users of a distributed system the convenience of providing the entire image hierarchy on a single storage medium permitting images containing watermarks to be accessed without restriction for browsing and proofing, while the watermark removal requires knowledge and us of a controlled code.

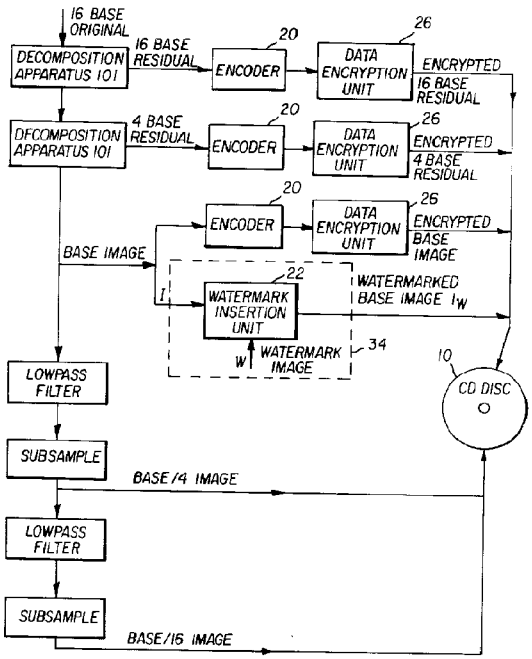


FIG 2

EP 0 651 554 A1

Cross-reference to Related Application:

The present application is related to U.S. Patent Application Serial No. 08/026,726, entitled "Method and Apparatus for Controlling Access to Selected Image Components In An Image Storage and Retrieval System" filed March 5, 1993, by P. W. Melnychuck and assigned to Kodak, the assignee of the present application.

Technical field Of The Invention

The present invention is related to the field of digital image processing and more particularly to methods and associated apparatuses for adding and removing a digital watermark to and from a selected image resolution and the preventing of unauthorized use of associated higher resolution digital image components.

Background Of The Invention

A number of hierarchical techniques for image coding have been described in the open technical literature and in various patents. Of particular relevance to the present invention are the following publications:

P. J. Burt and E. H. Adelson, "The Laplacian Pyramid As A Compact Code," IEEE Trans. Comm., COM-31, 532-540 (1983).

J. Seberry and J. Pieprzyk, "CRYPTOGRAPHY: An introduction to Computer Security" Prentice Hall, 1988 and the following patents:

U.S. Pat No, 4,969,204 entitled "Hybrid Residual-Based Hierarchical Storage And Display Method For High Resolution Digital Images In A Multiuse Environment," by Paul W. Melnychuck and Paul W. Jones, 1990.

U.S. Pat No, 5,048,111 entitled "Hybrid Subband-Based Hierarchical Storage And Display Method For High Resolution Digital Images In A Multiuse Environment," by Paul W. Jones and Paul W. Melnychuck, 1991.

The publication by Burt, et al. teaches an encoding method for images termed the Laplacian pyramid, the Burt pyramid, or the residual pyramid. In this technique, the original image is lowpass filtered, and this lowpass image is subsampled to take advantage of its reduced bandwidth to provide an image of reduced dimension. This process of lowpass filtering and subsampling is repeated three times to generate a hierarchical structure, or pyramid of images of successively smaller dimensions. The total number of resolution levels are created depending on the application. Each lowpass image in this pyramid is then expanded to the dimensions of the next higher level by upsampling (inserting zeros) and filtering to form a prediction image for that level. This prediction image is subtracted from its corresponding lowpass image in a subtractor to generate difference, or residual, images. The residual images corresponding to the levels of the lowpass pyramid form another pyramid which is termed the Laplacian, Burt, or residual pyramid. This technique is motivated by the fact that the residual images have a reduced variance and entropy compared to the original or lowpass images and may be quantized and entropy encoded to provide efficient storage of the data. Reconstruction is performed by interpolating the decoded lowpass image at the bottom of the lowpass pyramid and adding in the corresponding decoded residual to generate the next level in the lowpass pyramid. This process is iterated until the original image size is reached. A progressive improvement in reconstructed image quality and resolution can thus be obtained by displaying the reconstructed lowpass filtered image at each level of the pyramid. Note that errors introduced in the encoding process are propagated from one level to the next higher level in the decoding process.

The patent to Melnychuck and Jones (U. S. Pat. No. 4,969,204) teaches a modification of the Burt pyramid scheme by extending the lowpass pyramid structure to include one or more lowpass filtered images of successively smaller dimensions beyond the set described by Burt, et al. The advancement in the method of Melnychuck and Jones is that the residual pyramid is not extended to include these corresponding extended smaller dimensions. Hence, the Melnychuck and Jones pyramid contains the Burt pyramid plus additional lowpass filtered images of smaller dimensions. In a hierarchical image storage and retrieval system, the additional lowpass filtered images of smaller dimension can be retrieved directly, without interpolation and addition of residual components. In the context of the present invention, the Melnychuck and Jones pyramid provides for low resolution images that can be used for browsing or proofing. The use of these additional low resolution images for browsing and proofing means that the customer may use a simple retrieval mechanism and need not possess a more complex and hence, more expensive retrieval device that would be used to decode the higher resolution components of the pyramid. Of course, higher resolution images requiring interpolation and residual addition may be used for browsing and proofing as well.

A hierarchical image processing method will be described for the addition and removal of digital watermarks in selected image components, and for the restriction of selected high resolution image components from unauthorized use. An image hierarchy is constructed in the context of a multi-resolution environment whereby the user has the option of selecting the type of display medium and the desired resolution of this display medium. In particular, two types of display media are considered: video monitors and color hard copies, although photographic, thermal imaging,

and other types are also of interest. In Fig. 1 a prior art technique for decomposing, storing, recomposing, and displaying, a digital image using a hierarchical process is shown. An original digital image is decomposed to provide image versions at various resolutions to allow for the display of an HDTV quality image on video, an NTSC quality image with PAL/SECAM compatibility on video, one or more sub-NTSC quality images on video for overviews and browsing, and a very high quality image on color hard copy. Intermediate to the decomposition and recomposition steps, generally are inserted an encoding step, to compress the data for storage which in turn requires a decoding step when the data is read from storage.

Summary Of The Invention

The present invention places a digital watermark in a selected image resolution component and the means to remove it in an additional image component termed a watermark removal component. Encryption of the watermark removal component is used to prevent use of the image for the generation of unauthorized high quality color hard copy. A watermark is a form of graphic overlay that may contain a copyright notice or information regarding the restricted use of the image. In a distributed image system it is common to deliver an image of compromised image quality for purposes of browsing or proofing. A compromised rendition of the image is commonly distributed to prevent full utility or fulfillment of the image without proper payment for the service that generated the image. The term browsing refers to the process of image selection from a plurality of images based on some user-defined criterion. Such is the case when a user may select an image from a catalog of images depicting a particular object. The term proofing refers to the process of image selection based on the degree of desirability of a given image from a plurality of images. Such is the case when a professional portrait photographer distributes a plurality of images to a customer for selection and approval. The terms watermark, browsing and proofing described herein are not limited to the examples described above.

Upon selection of the desired image by the customer, the professional delivers a high quality rendition of the image, most often in the form of a high quality color hard copy. At all times the professional possesses the sole means of generating the high quality hard copy. In a conventional photographic system the means would be the original negatives of the images; in a digital hierarchical system according to the present invention, the means are higher resolution residual components.

In a digital imaging system, and in particular one that includes a hierarchical form of digital storage and retrieval, the professional may use a suitable digital storage medium such as a CD for the distribution of proofs. In an unrestricted environment, the customer may choose a desired image resolution from the hierarchy for the purposes of browsing, proofing, or hard copy fulfillment. In those instances where it is desirable for the professional to deliver the digital storage medium containing the entire image hierarchy to the customer; it is also most economical to record the entire image hierarchy once onto the digital storage medium and avoid having to make a second copy containing only low resolution components for distribution. However, it is also desirable to restrict the use of selected high resolution components for the purpose of full image quality fulfillment until payment has been received. The professional may choose to provide low resolution image components for browsing or proofing, while maintaining restriction of the higher resolution components. Alternatively, he may be required to deliver a proof of high resolution. Such is the case when the image content contains information of small detail and the rendition of this detail is subject to approval via the proof. With traditional photographic prints, the professional may place a stamp, or watermark on a strategic location on the print, so as to render the print useless from a fulfillment point of view. Note with digital images that fulfillment may mean high quality video at NTSC/PAL/SECAM, HDTV, or hard copy. In the present invention, the professional places a digital rendition of the watermark on a selected image component. The removal of the watermark is done through an additional image component containing the reverse of the watermark. The customer, having possession of the digital storage medium CD would possess the means for generating his own high quality hard copy when authorized by the professional. Upon payment to the professional, the professional or his agent provides to the customer the information necessary to remove the watermark for full image quality fulfillment. In the present invention, that information would be an authorization code, key, or password that would be inputted to the image processing system accessing the storage medium, to unlock the restricted high resolution components. An advantage of this technique is that the customer may possess all information pertinent to generating high quality hard copy without the need to physically return to the professional for additional image components.

It may additionally be desirable to use some form of hierarchical image representation for the purpose of browsing or proofing in a distributed system because the hierarchy naturally provides a plurality of resolutions, and hence levels of image quality, from which to choose the proof image. No additional operation of compromising the image is necessary; the professional simply chooses at what resolution level(s) he wants to restrict access.

Systems that use a hierarchical structuring of the image data have not been employed in the past for

distribution purposes because of the lack of means to simultaneously provide low resolution components for browsing and proofing, while offering restricted access to the remaining hierarchical components for full quality image copy. Additionally, the means to generate and remove a digital watermark in a hierarchical image structure had not been previously considered.

The present invention permits the advantages of hierarchical image decomposition to create a series of residual components, direct retrieval of the additional low resolution images according to the Melnychuck and Jones pyramid, the addition and removal of a digital watermark in a selected image resolution component, and prior art encryption methods applied to the watermark removal component and the residuals, to provide for a system of browsing, proofing and restriction of the high resolution image components suitable in a distributed image system. It is assumed that the residual components and the watermark removal component are symbol encoded using the encoder box 20 in Figures 2 and 4 into a binary string of 1's and 0's either via fixed-length coding techniques (where a binary code word of a fixed-length is assigned to each symbol) or variable-length encoding techniques such as Huffman coding or arithmetic coding. The residual data may also be quantized prior to encoding, or it may be encoded in a lossless manner, i.e., without quantization. Data encryption box 26 is applied to the watermark removal component and if desired, also to the encoded quantized (or non-quantized) residual data. It is assumed that the encryption process is reversible. Hence, the decryption box 28 provides the exact data prior to data encryption.

In one embodiment of the invention a storage medium is called for having stored therein at least one low resolution digital image and at least one high resolution digital image, with said high resolution digital image encoded with a watermark that requires an authorization code for removal.

From the foregoing, it can be seen that it is a primary object of the present invention to provide a method and associated apparatus for storing and controllably retrieving digital images stored in a hierarchical format on a suitable digital storage distribution medium that allows the originator of the distribution medium to distribute the medium containing the entire image hierarchy and a controllably removable watermark for the purpose of retrieving low resolution images for browsing or proofing without compromising the originator's need to withhold the means for creating hard copies of the images without the watermark.

It is another object of the present invention to provide the means for controllably inserting and removing a watermark for a digital image.

It is another object of the present invention to provide the means for compromising a selected image component of a hierarchical formatted digital image by adding a digital watermark to the selected image component, and recording the selected image component containing the watermark as part of the image hierarchy on a digital storage distribution medium.

In association with a digital image, it is another object of the present invention to provide a means for creating a watermark removal component, and for controllably restricting access to the watermark removal component.

It is another object of the present invention to provide a means for affixing a watermark to a digital image and for controllably removing the watermark.

The above and other objects of the present invention will become more apparent when taken in conjunction with the following description and drawings wherein like characters indicate like parts and which drawings form a part of the present description.

Brief Description Of The Drawings

Fig. 1 is a block diagram illustrating the prior art Melnychuck and Jones hierarchical storage and display method.

Fig. 2 is a functional block diagram illustrating a hierarchical image decomposition technique incorporating a watermark insertion into an image component.

Fig. 3 is a functional block diagram illustrating a reconstruction technique for reconstructing the images decomposed by the system of Fig. 2.

Fig. 4 is a functional block diagram of another hierarchical image decomposition technique incorporating a watermark insertion into an image component.

Fig. 5 is a functional block diagram illustrating a reconstruction technique for reconstructing the images decomposed by the system of Fig. 4.

Detailed Description of the Invention

In the following description of the preferred embodiments, it will be assumed that the highest resolution of the image hierarchy is composed of 3072 x 2048 pixels and that this resolution is adequate to produce photographic quality originals on an appropriate digital output device. It is also assumed that a moderately high resolution level of the hierarchy composed of 1536 x 1024 pixels is adequate to generate a high quality HDTV display, or a small-sized photographic quality print on an appropriate digital output device. It is also assumed that the lowest resolution levels of 192 x 128 pixels, 384 x 256 pixels, and 768 x 512 pixels are generated and stored onto a digital storage medium such as a CD. These resolution levels are provided to give the reader an insight as to the operation of one or more embodiments of the invention

with the understanding that other resolutions or arrangements may be chosen to suit specific needs without detracting from the teachings of the present invention.

Referring now to Fig. 2, a hierarchical residual decomposition technique, for decomposing a 16BASE original image to form a 16BASE residual, a 4BASE residual, a BASE, a BASE/4, and a BASE/16 image, incorporating the teachings found substantially in Fig. 7 of the patent to Melnychuck and Jones (U. S. Pat. No. 4,969,204), in combination with the present invention is shown. The BASE image is processed in box 34 to incorporate a watermark and to provide a watermarked BASE image.

An example of a watermark insertion box 34 is given by the watermark insertion unit 22 whereby a watermark image W is combined with the input image I to create a watermarked image I_W. In this example, it is assumed that the input image I and the watermark image W are of the same size and the same bit-depth. For example, if the input image I is an 8-bit image representing the luminance component of a color image, the watermark image W would also be an 8-bit image. Similarly, the watermarked image I_W would have the same size and each pixel value would be represented with 8 bits. An example of a watermark insertion unit 22 is one where the input image I and the watermark image W are combined according to the following equation to create the watermarked image I_W

$$I_W(i,j) = I(i,j) + \alpha\, W(i,j)$$

Where (i,j) denotes the two-dimensional location of the pixels in the image and the operation is performed for all the pixels in the input image. The watermark image W is prepared by the originator of the storage medium and may contain the logo of the originator or any other pattern that the originator may wish to use as a watermark. The parameter α, which can be either positive or negative, controls the watermark contrast and is also selected by the originator and can vary from one image to another. Larger magnitudes of α would, in general, create a higher contrast watermark. Also, to guarantee that the watermarked image I_W has the same bit-depth as the input image I, the watermarked image I_W is clipped to the same range as the input image. For example, for an 8-bit image with pixel values in the range of 0 to 255, for every pixel location (i,j), the value of $I_W(i,j)$ is clipped to 255 if the result of the above equation exceeds 255 and is set to zero if that result is less than zero. It should be noted that this example illustrates only one method of implementing the watermark insertion box 34 and the originator of the storage medium may incorporate any other method to generate a watermark that creates the desired effect of inhibiting the use of the image.

The BASE/16, BASE/4, and watermarked BASE images are stored on the digital storage medium 10 in direct (unencrypted) form. The BASE image, which in this case serves as the watermark removal record, is encrypted in the data encryption unit 26. The data encryption unit 26 consists of either a private-key data encryption algorithm (also referred to as symmetric data encryption algorithm) or a public-key data encryption algorithm (also referred to as asymmetric data encryption algorithm) both of which have been explained in the prior art and in the reference book by Seberry and Pieprzyk cited before. Examples of private-key encryption algorithms that can be used in the data encryption unit 26 are either block ciphers such as the Data Encryption Standard (DES) which uses a 56-bit key and operates on blocks of data of length 64 bits at a time, or a stream cipher algorithm such as RC-4, a commercially available encryption software that uses a 40-bit key component. The encrypted BASE image is also stored on the storage medium 10. The 4BASE and 16BASE residual components are also stored on the digital storage medium 10 either in direct (unencrypted) form or in encrypted form depending on the level of security desired by the application. In the case that the encryption of any or all of the residual data are needed, either the same key used in encrypting the BASE image is used or a separate key is used. The use of multiple encryption keys provides the originator of the storage medium with more flexibility in controlling the access to the various resolutions of the image hierarchy.

For browsing or proofing, a procedure illustrated by Fig. 3 is employed. A user retrieves the BASE/16, BASE/4, or watermarked BASE image directly without decryption from the digital storage medium 10. Upon authorization, the user inputs a decryption key(s) to the data decryption unit 28 to allow the decryption of the original BASE image (and the residuals) to be performed. An example of a data decryption unit 28 is a software implementation of a decryption algorithm corresponding to the reverse operation of the encryption algorithm employed in the data encryption unit 26. One example of a set of encryption/decryption algorithms is the Data Encryption Standard (DES) which has been explained in full detail in the reference book by Seberry *et al* mentioned before. Note that the decryption key(s) must be provided by the originator of the storage medium. Upon the decryption of the BASE image and the residual components, these components can be used to arrive at full image quality fulfillment.

In a second embodiment, illustrated in Fig. 4, the 16BASE image is decomposed by decomposition apparatus 101 into a residual pyramid consisting of the 16BASE, 4BASE, and BASE. The BASE image is further decomposed to create the BASE/4 and BASE/16 images, through low pass filtering and subsampling. BASE /4 and BASE/16 are not part of the residual pyramid and hence they are available directly for display on a monitor.

A watermark, as described in the previous em-

bodiment in Fig. 2, is inserted in the BASE image in box 34 to arrive at a watermarked BASE image. This watermarked BASE image is then interpolated to the size of the 4BASE image using linear interpolation as indicated by the interpolator box 24. A difference is formed in subtractor 32 between the original 4BASE image and the interpolated watermarked BASE image to form a modified 4BASE residual that serves as the watermark removal record. The difference in this embodiment versus the first embodiment is that the watermark removal record is the modified 4BASE residual instead of the BASE image. This modified 4BASE residual is encrypted using the data encryption unit 26 as described before and is then stored on the storage media 10 along with the BASE/16, BASE/4, and watermarked BASE image in direct (unencrypted) form. Finally, the 16BASE residual data is stored on the digital storage medium either in direct or encrypted form depending on the application.

For browsing or proofing, the system of Fig. 5 is employed. The user retrieves the BASE/16, BASE/4, or watermarked BASE image directly without decryption from the digital storage medium 10. Upon authorization, the user inputs the decryption key to the data decryption unit 28 to allow the decryption to be performed to generate the modified 4BASE residual. The watermarked BASE image is interpolated using linear interpolation and is added to the decrypted modified 4BASE residual in the reconstruction apparatus 210 to recover the original 4BASE image. If the residuals have not been quantized, the 4BASE image can be exactly recovered. In the case where the residuals have been quantized, some discrepancy between the original 4BASE image and the 4BASE image recovered according to the above scheme would exist. The degree of this discrepancy would depend on the coarseness of the quantizer employed in the quantization of the residual components. Note that the decryption key must be provided by the originator of the storage medium.

It is to be understood that in some instances it may be desirable to place a watermark upon the low resolution images to control their access.

While there has been shown what are considered to be the preferred embodiments of the invention, it will be manifest that many changes and modifications may be made therein without departing from the essential spirit of the invention. It is intended, therefore, in the annexed claims, to cover all such changes and modifications as may fall within the scope of the invention.

Parts List:

10 Digital storage medium (CD-Disc)

20 Encoder
22 Watermark insertion unit
24 Interpolator
26 Data encryption unit
28 Date decryption unit

30 Decoder
32 Subtractor
34 Watermark insertion box

101 Decomposition apparatus

201 Reconstruction apparatus

Claims

1. A storage medium having stored therein at least one low resolution digital image and at least one high resolution digital image, with said high resolution digital image encoded with a watermark that requires an authorization code for removal.

2. The storage medium according to claim 1 and further having stored thereon at least one additional high resolution digital image that is not encoded with a watermark and is accessed with the authorization code in place of the high resolution digital image encoded with the watermark.

3. A storage medium having stored therein at least one low resolution digital image and at least one high resolution digital image in the form of a BASE image, residual image components and a watermark component, with said low resolution digital image, said BASE image or said high resolution image formed by the combination of the BASE image with said residual image components and a watermark component being accessible without an authorization code.

4. The storage medium of claim 3 in combination with an authorization code to remove the watermark component from an accessed high resolution image.

5. A system for controlling the uncompromised use of a high resolution digital image stored on a storage medium as BASE and residual components, comprising:

means for encrypting the residual components stored on said storage medium using a watermark code;

means for accessing the BASE and encrypted residual components;

means for combining the accessed BASE and residual components to reconstruct the high resolution digital image with the watermark code; and

means for authorizing the removal of the watermark code.

6. A system for controlling the uncompromised use of a high resolution digital image comprising:

means for forming a hierarchy of lower resolution digital images from the high resolution digital image;

means for forming residual images that are a function of differences between adjacent images in the hierarchy of lower resolution digital images;

means for encrypting at least one of the formed residual images with a watermark code;

storage means for storing the formed hierarchy of lower resolution images and the at least one encrypted residual image;

means for reconstructing high resolution images by accessing and combining a lower resolution image with a residual image;

means for displaying of the at least one encrypted residual image with the watermark; and

means for controllably removing the watermark code to permit an uncompromised use of the high resolution digital image.

7. A recording medium having stored thereon a plurality of digital images with each of the digital images being comprised of a low resolution digital image component and at least one residual digital image component which is combinable with the low resolution digital image component to form a higher resolution digital image incorporating a watermark which is removable with an authorization code.

8. A method for controlling the use of a digital image stored on a storage medium in a hierarchical form comprised of a BASE image and at least one residual image component, comprising the steps of:

a) associating a watermark with said at least one residual image component;

b) permitting access to the BASE image for low resolution viewing of the digital image;

c) combining the BASE image with the at least one residual image component and an associated watermark to form the digital image for viewing, printing and/or storing; and

d) controllably providing a watermark removal code to remove the watermark from the formed digital image of step c.

9) A storage medium having stored thereon at least one digital image encoded with a watermark that requires an authorization code for removal.

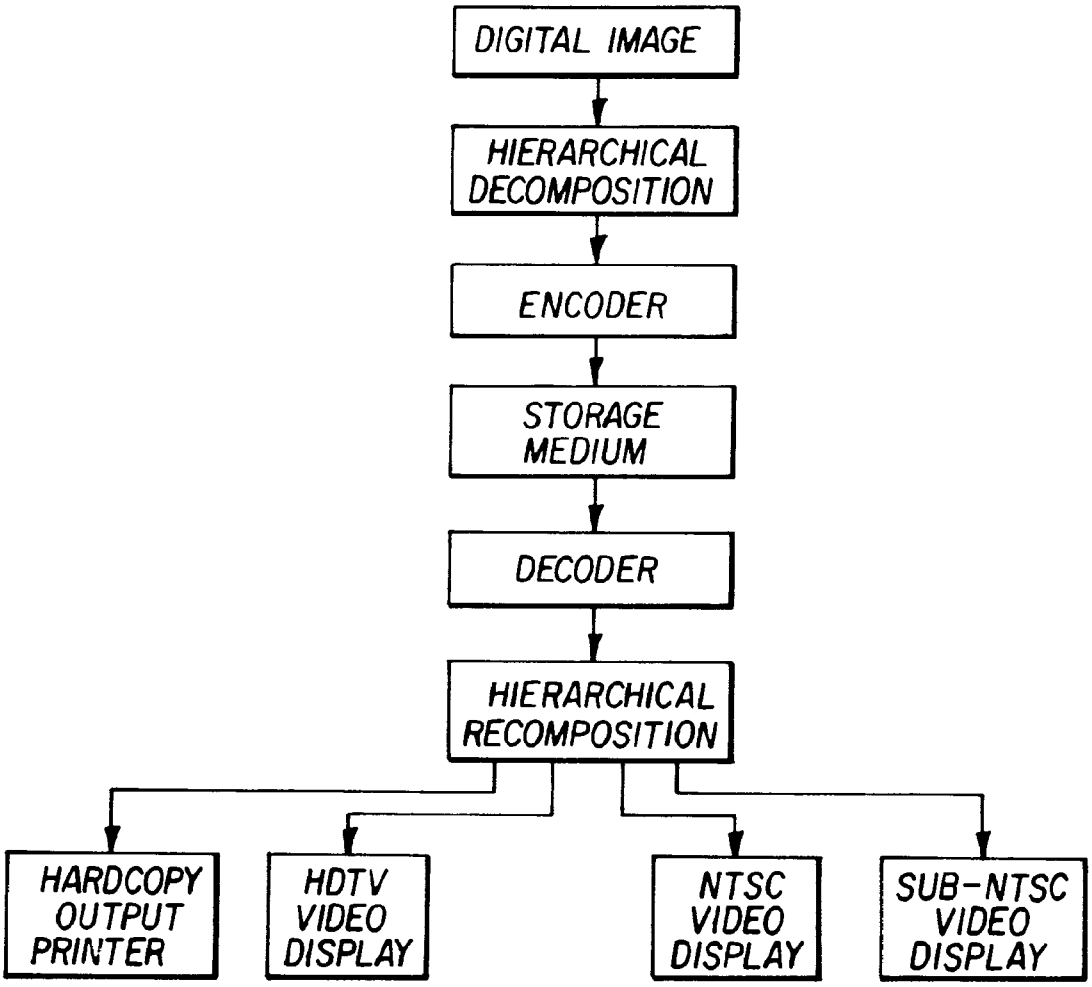


FIG. 1
(prior art)

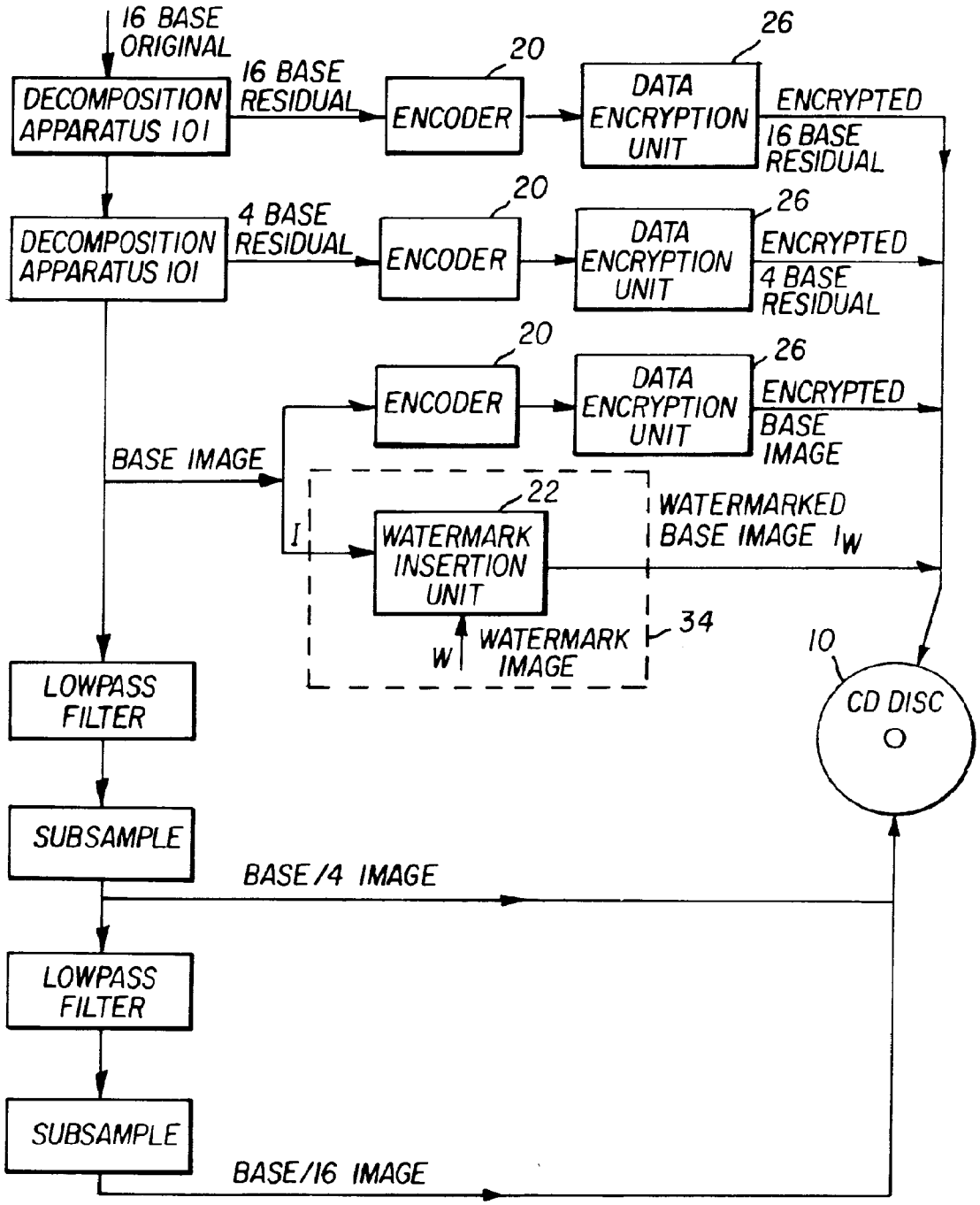

16 BASE ORIGINAL
DECOMPOSITION APPARATUS 101
16 BASE RESIDUAL
20
ENCODER
26
DATA ENCRYPTION UNIT
ENCRYPTED 16 BASE RESIDUAL
DECOMPOSITION APPARATUS 101
4 BASE RESIDUAL
20
ENCODER
26
DATA ENCRYPTION UNIT
ENCRYPTED 4 BASE RESIDUAL
20
ENCODER
26
DATA ENCRYPTION UNIT
ENCRYPTED BASE IMAGE
BASE IMAGE
I
22
WATERMARK INSERTION UNIT
WATERMARKED BASE IMAGE I_W
W
WATERMARK IMAGE
34
10
CD DISC
LOWPASS FILTER
SUBSAMPLE
BASE/4 IMAGE
LOWPASS FILTER
SUBSAMPLE
BASE/16 IMAGE

FIG. 2

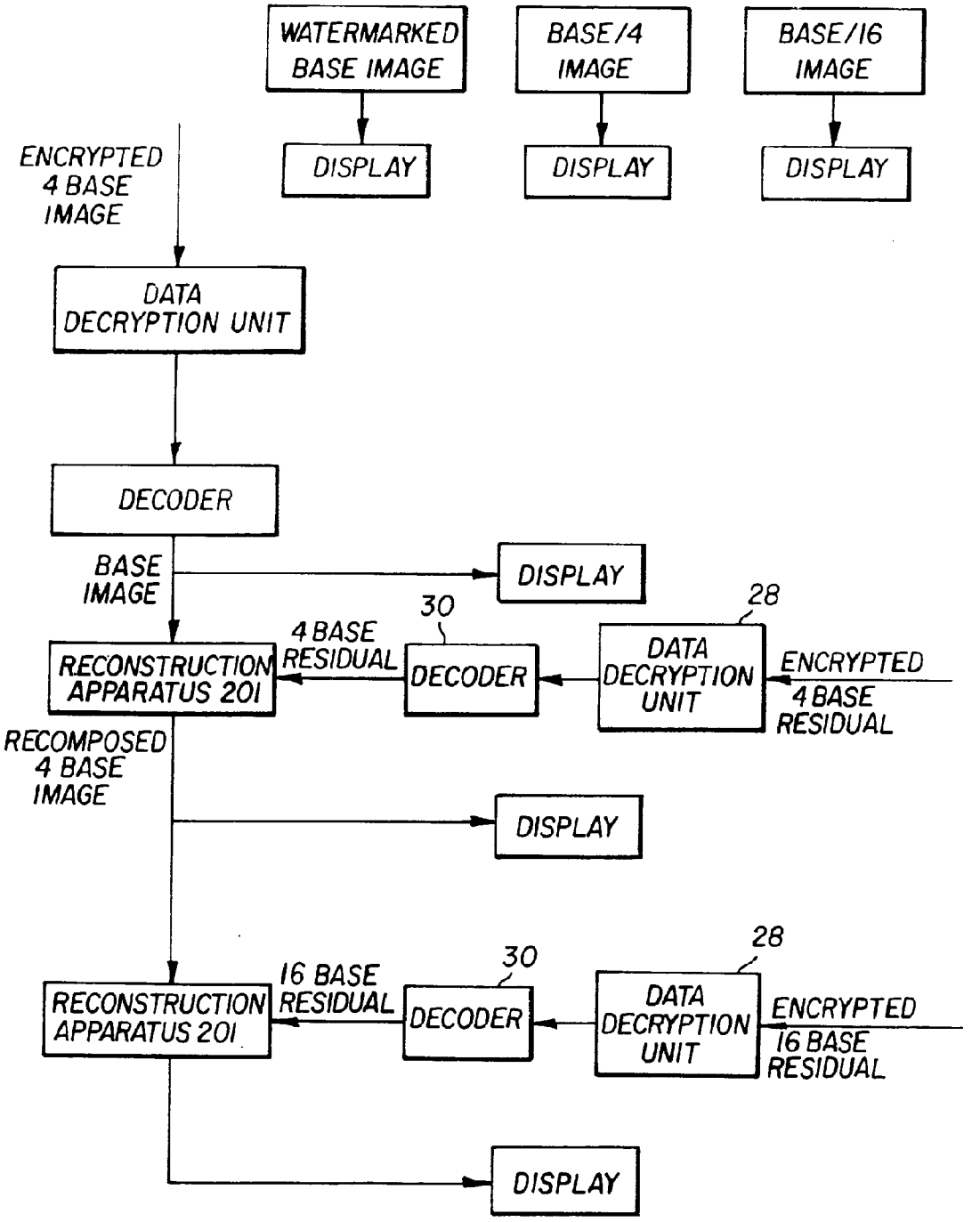

WATERMARKED BASE IMAGE
BASE/4 IMAGE
BASE/16 IMAGE
DISPLAY
DISPLAY
DISPLAY
ENCRYPTED 4 BASE IMAGE
DATA DECRYPTION UNIT
DECODER
BASE IMAGE
DISPLAY
30
28
4 BASE RESIDUAL
RECONSTRUCTION APPARATUS 201
DECODER
DATA DECRYPTION UNIT
ENCRYPTED 4 BASE RESIDUAL
RECOMPOSED 4 BASE IMAGE
DISPLAY
30
28
16 BASE RESIDUAL
RECONSTRUCTION APPARATUS 201
DECODER
DATA DECRYPTION UNIT
ENCRYPTED 16 BASE RESIDUAL
DISPLAY

FIG. 3

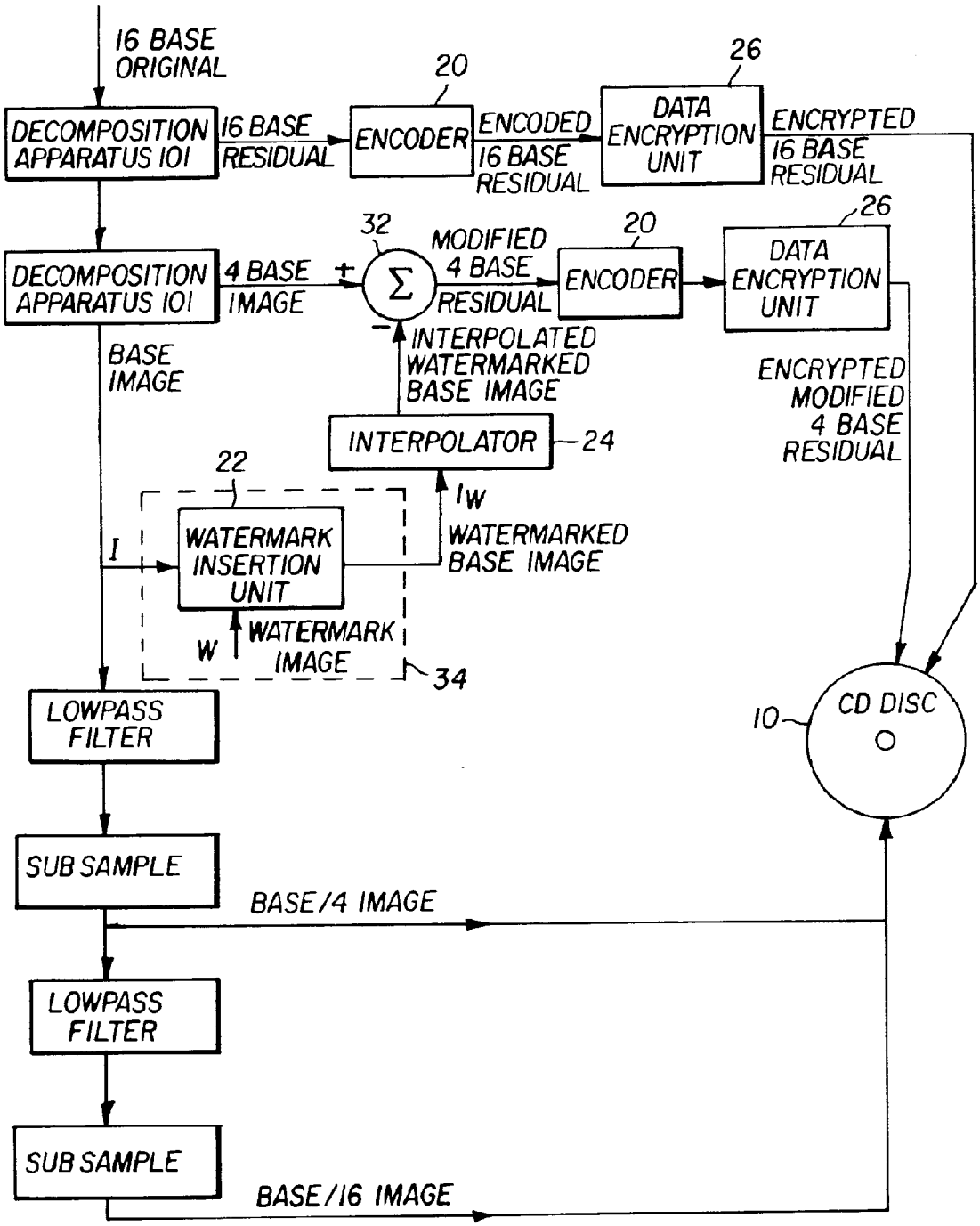

16 BASE ORIGINAL
DECOMPOSITION APPARATUS 101
16 BASE RESIDUAL
20
ENCODER
ENCODED 16 BASE RESIDUAL
26
DATA ENCRYPTION UNIT
ENCRYPTED 16 BASE RESIDUAL
DECOMPOSITION APPARATUS 101
4 BASE IMAGE
32
Σ
MODIFIED 4 BASE RESIDUAL
20
ENCODER
26
DATA ENCRYPTION UNIT
BASE IMAGE
INTERPOLATED WATERMARKED BASE IMAGE
ENCRYPTED MODIFIED 4 BASE RESIDUAL
INTERPOLATOR
24
22
I_W
I
WATERMARK INSERTION UNIT
WATERMARKED BASE IMAGE
W
WATERMARK IMAGE
34
LOWPASS FILTER
10
CD DISC
SUB SAMPLE
BASE/4 IMAGE
LOWPASS FILTER
SUB SAMPLE
BASE/16 IMAGE

FIG. 4

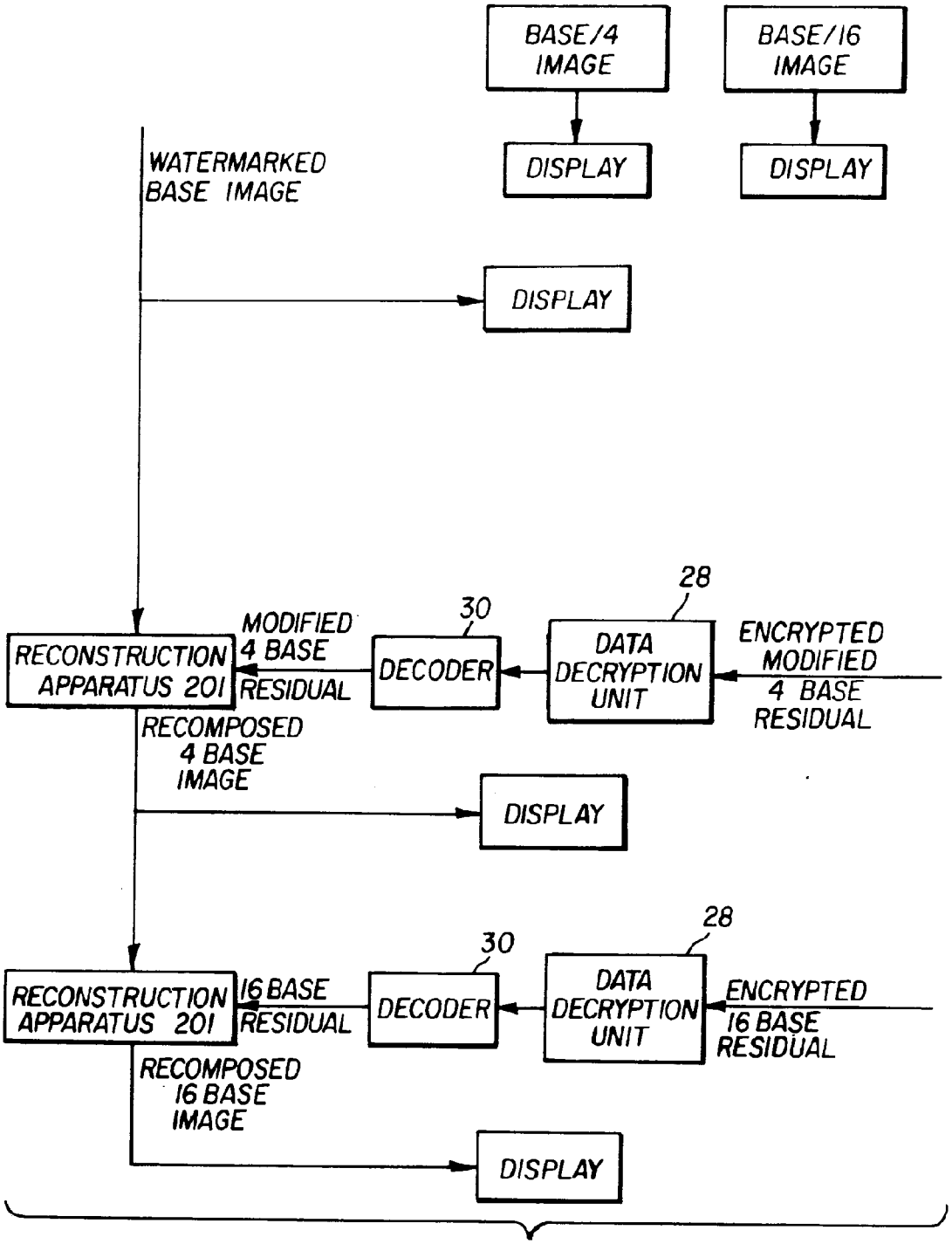

BASE/4 IMAGE
BASE/16 IMAGE
DISPLAY
DISPLAY
WATERMARKED BASE IMAGE
DISPLAY
28
30
MODIFIED 4 BASE RESIDUAL
RECONSTRUCTION APPARATUS 201
DECODER
DATA DECRYPTION UNIT
ENCRYPTED MODIFIED 4 BASE RESIDUAL
RECOMPOSED 4 BASE IMAGE
DISPLAY
28
30
16 BASE RESIDUAL
RECONSTRUCTION APPARATUS 201
DECODER
DATA DECRYPTION UNIT
ENCRYPTED 16 BASE RESIDUAL
RECOMPOSED 16 BASE IMAGE
DISPLAY

FIG. 5

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 94 42 0293

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.6)
D,Y	US-A-4 696 204 (MELNYCHUCK ET AL) * the whole document * ---	1-9	H04N1/21 G06F1/00
Y	ELECTRONICS AND COMMUNICATIONS IN JAPAN, vol.73, no.5, May 1990, NEW YORK, US; pages 22 - 33 N.KOMATSU ET AL 'A Proposal on Digital Watermark in Document Image Communication and Its Application to Realizing a Digital Signature' * figures 1-5 * * page 22, left column, line 1 - page 27, left column, line 23 * ---	1-9	
P,Y	EP-A-0 614 308 (EASTMAN KODAK) * abstract; figure 2 * * column 4, line 52 - column 5, line 4 * -----	2	

TECHNICAL FIELDS SEARCHED (Int.Cl.6)
H04N
G06F

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	30 January 1995	Powell, D

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

(19)日本国特許庁（ＪＰ）　(12) **公開特許公報（A）**　(11)特許出願公開番号

特開平7－66829

(43)公開日　平成7年(1995)3月10日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ	技術表示箇所
Ｈ04Ｌ 12/54				
Ｇ06Ｆ 13/00	351 Ｇ	7368－5B		
Ｈ04Ｌ 12/58				
		8732－5K	Ｈ04Ｌ 11/20	101 Ｂ

審査請求　未請求　請求項の数5　ＯＬ　（全 8 頁）

(21)出願番号　特願平5－162529

(22)出願日　平成5年(1993)6月30日

(71)出願人　000001258
川崎製鉄株式会社
兵庫県神戸市中央区北本町通1丁目1番28号

(72)発明者　斎藤　洋
東京都千代田区内幸町2丁目2番3号　川崎製鉄株式会社東京本社内

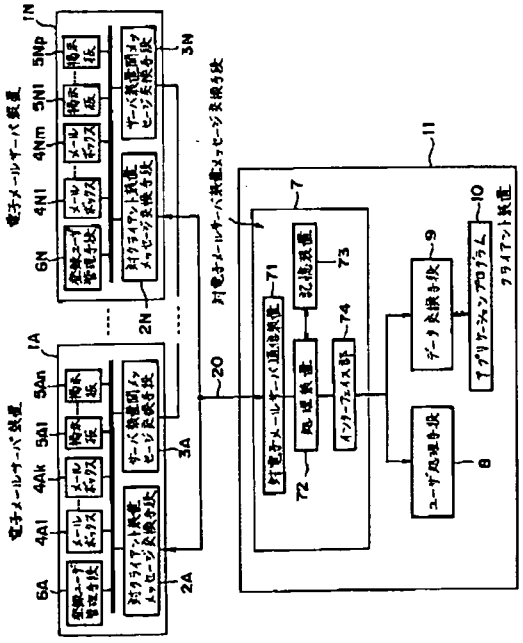
(74)代理人　弁理士　谷　義一　（外1名）

(54)【発明の名称】　電子メール多重化システムおよび該システムにおける通信制御方法

(57)【要約】

【構成】　多重化された電子メールサーバ装置（1A－1N）がそれぞれ有する、メールボックス（または、電子掲示板）の内容を、相互に複写する機能を、各電子メールサーバ装置にもたせ、クライアント装置（11）内の対電子メールサーバ装置メッセージ交換手段（7）が、稼働中の電子メールサーバ装置の1台を自動的に選択して、両者を接続するようにした。また、電子メールサーバ装置からの情報に基づいて、最も負荷の少ない電子メールサーバ装置をクライアント装置が選択する。

【効果】　複数の電子メールサーバ装置のうち、少なくとも1台が稼働状態であれば、サービスが継続される。多重化された電子メールサーバ装置間の負荷分散が可能である。



1

【特許請求の範囲】

【請求項１】　多重化された電子メールサーバ装置とクライアント装置とがネットワークを介して接続された電子メールシステムにおいて、

前記電子メールサーバ装置は、

メールボックスまたは電子掲示板等のメッセージ蓄積手段と、

登録されたユーザの管理をする登録ユーザ管理手段と、

前記クライアント装置とのメッセージの送受信を行う対クライアント装置メッセージ交換手段と、

前記電子メールサーバ装置間でのメッセージの交換を行うサーバ装置間メッセージ交換手段とを備え、

前記クライアント装置は、

使用者またはアプリケーションプログラムとのインターフェイスをとる手段と、

前記電子メールサーバ装置とのメッセージの送受信を行う対電子メールサーバ装置メッセージ交換手段とを備え、

前記サーバ装置間メッセージ交換手段は、複数の電子メールサーバ装置間で、前記メッセージ蓄積手段の内容を複写し、

前記対電子メールサーバ装置メッセージ交換手段は、前記電子メールサーバ装置からの情報に基づいて、前記電子メールサーバ装置の中から、稼働中の装置を通信相手として選択することによって、電子メールシステムの無停止を実現することを特徴とする電子メール多重化システム。

【請求項２】　請求項１において、前記対電子メールサーバ装置メッセージ交換手段は、前記電子メールサーバ装置からの情報に基づいて、負荷の少ない電子メールサーバ装置を選択することを特徴とする電子メール多重化システム。

【請求項３】　多重化された電子メールサーバ装置とクライアント装置とがネットワークを介して接続された電子メールシステムにおいて、

前記電子メールサーバ装置間でメッセージの交換を行い、複数の電子メールサーバ装置間で、前記メッセージを複写する第１の過程と、

前記クライアント装置が、前記電子メールサーバ装置からの情報に基づいて、前記電子メールサーバ装置の中から、稼働中の装置を通信相手として選択する第２の過程とを具備することを特徴とする電子メール多重化システムにおける通信制御方法。

【請求項４】　請求項３において、前記第２の過程は、

前記クライアント装置が前記電子メールサーバ装置に対して自局のＩＤを一斉送信する過程と、

該送信に応じた複数の電子メールサーバ装置の中から一つの電子メールサーバ装置を選択する過程と、

選択した電子メールサーバ装置に接続要求する過程とを具備することを特徴とする電子メール多重化システムにおける通信制御方法。

2

【請求項５】　請求項４において、前記クライアント装置が、通信相手の電子メールサーバ装置の故障を検出した場合には、前記一斉送信に応じた複数の電子メールサーバ装置の中から稼働中の他の装置を選択して、接続要求を送信することを特徴とする電子メール多重化システムにおける通信制御方法。

【発明の詳細な説明】

【０００１】

【産業上の利用分野】本発明は、多重化された電子メールサーバ装置を備えた電子メール多重化システムおよび該システムにおける通信制御方法に関する。

【０００２】

【従来の技術】従来の電子メールシステムとしては、特開平４－２７３６３６号公報に記載された分散掲示板方式が知られている。これは、電子メールシステムにおいて、複数のサーバ装置に掲示板のコピーを配置することにより、掲示物の読み出し要求が１つのサーバ装置に集中することを防止し、応答性を向上させようとしたものである。

【０００３】この従来の電子メールシステムにおいては、電子メールサーバ装置とクライアント装置とが、ネットワークを介して、固定的に接続されていた。すなわち、特定の利用者が、電子メールシステムから必要なデータを得る場合は、特定の１台の電子メールサーバ装置からそのデータを得ていた。なお、この種のデータを利用するものとしては、一般に、利用者とアプリケーションプログラムとがあるが、以下の説明においては、これらを総称して、「サービス受益対象」と呼ぶこととする。

【０００４】

【発明が解決しようとする課題】従来の電子メールシステムにおいては、特定のサービス受益対象と電子メールサーバ装置とが固定的に接続されているため、自分が接続された電子メールサーバ装置が停止した場合、サービス受益対象は、この電子メールシステムを利用することができなかった。このような事態を避けるためには、電子メールシステムの無停止サービスを実現しなければならない。具体的には、電子メールサーバ装置にフォールトトレラント型のコンピュータを適用したり、複数の電子メールサーバ装置を高度に結合し、多重化する方法がとられていた。これらの装置を実現するには、高度な制御技術と、多大な費用が要求された。

【０００５】さらに、同一のネットワーク上に複数の電子メールサーバ装置が稼働していても、特定の電子メールサーバ装置に、アクセスが集中してしまう可能性があった。

【０００６】本発明は、無停止で稼働する安価な電子メール多重化システムを提供することを目的とする。また、多重化された電子メールサーバ装置間の負荷分散を

可能にしたシステムを提供することを目的とする。

【0007】

【課題を解決するための手段】上記課題を解決するために、本発明は、多重化された電子メールサーバ装置に、それぞれが有するメールボックス（または、電子掲示板）の内容を、相互に複写する機能をもたせるとともに、クライアント装置が、稼働中の電子メールサーバ装置の1台を自動的に選択して接続するようにした。すなわち、本発明では、電子メールサーバ装置の切り替えを、クライアント装置が行うようにした。

【0008】また、電子メールサーバ装置からの情報に基づいて、最も負荷の少ない電子メールサーバ装置をクライアント装置が選択することによって、複数の電子メールサーバ装置間の負荷のバランスをとっている。

【0009】

【作用】本発明によれば、複数の電子メールサーバ装置のうち、少なくとも1台が稼働状態であれば、クライアント装置内に配置されている対電子メールサーバ装置メッセージ交換手段が、稼働中の電子メールサーバ装置の中の1台を自動的に選択して接続を行う。

【0010】さらに、クライアント装置が接続する際に、電子メールサーバ装置側からの情報に基づいて、負荷の最も少ない電子メールサーバ装置に接続することによって、多重化された電子メールサーバ装置間の負荷分散を可能にした。

【0011】

【実施例】次に、本発明の実施例を詳細に説明する。

【0012】図1は、本発明の一実施例を示す図である。

【0013】図1において、電子メールサーバ装置1A－1Nは、メッセージを蓄積して、要求に応じてメッセージを要求元に転送する装置である。また、クライアント装置11は、サービス受益対象が、電子メールサーバ装置1A－1Nにサービスを要求するための装置である。

【0014】電子メールサーバ装置1A－1Nは、それぞれ、対クライアント装置メッセージ交換手段2A－2N、サーバ装置間メッセージ交換手段3A－3N、メールボックス4A1－4Ak－4N1－4Nm、掲示板5A1－5An－5N1－5Np、および登録ユーザ管理手段6A－6Nを備えている。

【0015】ここで、各メールボックスおよび掲示板は、メッセージ蓄積手段として機能するもので、サーバ装置間メッセージ交換手段3A－3Nを介して、相互に接続されている。登録ユーザ管理手段6A－6Nは、電子メールサーバ装置1A－1Nに接続を要求してきたサービス受益対象の認証を行うもので、内部にユーザIDとパスワードとを格納している。

【0016】クライアント装置11は、対電子メールサーバ装置メッセージ交換手段7と、ユーザ処理手段8と、データ交換手段9と、アプリケーションプログラム10とを備えている。ここで、対電子メールサーバ装置メッセージ交換手段7は、各電子メールサーバ装置との通信を制御する対電子メールサーバ通信装置71と、処理装置72と、プログラムおよび各種データを格納する記憶装置73と、インターフェイス部74とを有している。対電子メールサーバ通信装置71は、処理装置72からの要求によって、図2および図3に示す通信制御を行う。また、インターフェイス部74は、処理装置72とユーザ処理装置8およびデータ交換手段9とのインターフェイスをとる。

【0017】ユーザ処理手段8は、利用者からのメッセージや指示を受け取り、電子メールサーバ装置1A－1Nに対して、メッセージを転送したり、メッセージの転送を要求したり、また転送されてきたメッセージを利用者に供給する。電子メールサーバ装置との、これらのやり取りは、クライアント装置内の対電子メールサーバ装置メッセージ交換手段7、ネットワーク20、電子メールサーバ装置内の対クライアント装置メッセージ交換手段2A－2Nを介して行われる。

【0018】アプリケーションプログラム10は、クライアント装置11内に、必要に応じて配置されたもので、データ交換手段9および対電子メールサーバ装置メッセージ交換手段7を通して、電子メールサーバ装置1A－1Nと接続され、メッセージ交換を行う。

【0019】次に、電子メールサーバ装置1（1A－1N）の多重化について説明する。

【0020】ある電子メールサーバ装置内のメールボックス4および掲示板5は、サーバ装置間メッセージ交換手段3を通して、他の電子メールサーバ装置内のメールボックス4および掲示板5にコピーされる。このコピーは、ネットワークの構成や、メッセージの転送頻度などに応じて決められる。具体的には、夜間や休日など、装置が比較的すいている間に、一括コピーする方法や、メッセージがいずれかの電子メールサーバ装置に到着した時点で、ただちに、他の電子メールサーバ装置のメールボックスや掲示板にコピーする方法などがある。なお、コピーの数は、必ずしも同一である必要はなく、メッセージの重要性に応じて、変えてもよい。

【0021】次に、図2および図3を参照して、本実施例の動作を説明する。

【0022】図2は、接続時の手順を示すものであり、サービス受益対象（ユーザまたはアプリケーションプログラム）、対電子メールサーバ装置メッセージ交換手段7、および電子メールサーバ装置の3つに分けて、動作を示している。

【0023】サービス受益対象は、電子メールサーバ装置1からのサービスを受けるために、自身のIDおよびパスワードを、クライアント装置11に指示する（ステップS1－S4）。クライアント装置11のユーザ処理

5

手段8またはデータ交換手段9は、対電子メールサーバ装置メッセージ交換手段7に対して、ＩＤとパスワードとを引き渡した上で、電子メールサーバ装置1への接続を依頼する。このとき、対電子メールサーバ装置メッセージ交換手段7は、サービス受益対象のＩＤとパスワードとを内部的に記憶しておく。

【００２４】次に、対電子メールサーバ装置メッセージ交換手段7は、接続可能なすべての電子メールサーバ装置1に対して、サービス受益対象のＩＤと特定の指示とを、一斉送信する（ステップＳ５）。ここで、特定の指示とは、上記ＩＤをもつサービス受益対象にサービス提供可能な電子メールサーバ装置は、その電子メールサーバ装置の識別コードを返送するように、との指示である。

【００２５】一斉送信を受信した電子メールサーバ装置1Ａ－1Ｎは、受信したサービス受益対象のＩＤと、自分が管理する登録ユーザ管理手段6Ａ－6Ｎ内のデータとを照合し、一致するものがあれば、自分の識別コードを、対クライアント装置メッセージ交換手段2Ａ－2Ｎを経由して、問い合わせ元の対電子メールサーバ装置メッセージ交換手段7に返す（ステップＳ６－Ｓ９）。こうして、このサービス受益対象にサービスを提供できる電子メールサーバ装置の識別コードが、対電子メールサーバ装置メッセージ交換手段7に受信される（ステップＳ１０）。

【００２６】対電子メールサーバ装置メッセージ交換手段7は、サービスを提供できる電子メールサーバ装置のリストを作成し（ステップＳ１１）、接続すべき電子メールサーバ装置を選択する（ステップＳ１３）。この場合、各電子メールサーバ装置に対して、現在の負荷状況などの問い合わせをして（ステップＳ１２）、電子メールサーバ装置を決定してもよい。一方、サービス提供可能な電子メールサーバ装置が存在しない場合、対電子メールサーバ装置メッセージ交換手段7は、サービス受益対象に対して、サービス提供ができない旨の通知を行う（ステップＳ１４－Ｓ１５）。サービス受益対象は、この通知を受信して、接続が失敗であったことを認識し、処理を終了する（ステップＳ１６）。

【００２７】利用可能な電子メールサーバ装置が複数存在する場合、対電子メールサーバ装置メッセージ交換手段7は、あらかじめ定められた手順で、一台の電子メールサーバ装置を選択する。具体的には、次のような方法が考えられる。

【００２８】（1）　サービス可能なすべての電子メールサーバ装置に対して、処理負荷に関する問い合わせを行い（ステップＳ１２）、その時点で負荷が最も少ない電子メールサーバ装置を選択する。

【００２９】（2）　対電子メールサーバ装置メッセージ交換手段7に、最初に識別コードを返した電子メールサーバ装置を選択する。

6

【００３０】電子メールサーバ装置の選択が完了すると、対電子メールサーバ装置メッセージ交換手段7は、選択した電子メールサーバ装置に対して、ログイン要求、ユーザＩＤ、およびパスワードを送信する（ステップＳ１７）。電子メールサーバ装置は、これらを受信し（ステップＳ１８）、接続が可能か否かを確認のうえ（ステップＳ１９）、接続を実行する（ステップＳ２０）。接続完了後、あるいは接続が不可能な場合、電子メールサーバ装置は、対電子メールサーバ装置メッセージ交換手段7に対して、結果を通知する（ステップＳ２１）。

【００３１】対電子メールサーバ装置メッセージ交換手段7は、接続結果を受信し（ステップＳ２２）、接続が成功したときには、サービス開始をサービス受益対象に通知する（ステップＳ２３－Ｓ２４）。サービス受益対象は、このサービス開始通知を受信し、次の段階（サービス要求）に入る。一方、接続が失敗に終った場合は、ステップＳ１３に戻って、他の電子メールサーバ装置を選択し、この電子メールサーバ装置に接続要求を行う。指定した電子メールサーバ装置との接続に再度失敗した場合は、この電子メールサーバ装置を、サービス可能な電子メールサーバ装置のリストから外して、他の電子メールサーバ装置にもう一度接続要求を行う。

【００３２】このようにして、電子メールサーバ装置との接続に成功するか、あるいは、サービス可能な電子メールサーバ装置がなくなるまで、ステップＳ１３－Ｓ２３の手順を繰り返す。

【００３３】図3は、正常時のサービス要求の動作と、電子メールサーバ装置に障害が発生した場合の動作とを示すフローチャートである。

【００３４】電子メールサーバ装置との接続に成功すると、サービス受益対象は、対電子メールサーバ装置メッセージ交換手段7に対して、サービス要求を送信する（ステップＳ３１）。このサービス要求を受信すると、対電子メールサーバ装置メッセージ交換手段7は、電子メールサーバ装置に対して、サービス要求を送信する（ステップＳ３２）。サービス要求を受信した電子メールサーバ装置は、サービスを実行し、サービス内容を対電子メールサーバ装置メッセージ交換手段7に返送する（ステップＳ３３－Ｓ３５）。

【００３５】サービス内容を受信した対電子メールサーバ装置メッセージ交換手段7は、サービス内容をサービス受益対象に転送する（ステップＳ３６－Ｓ３７）。サービス受益対象はこのサービス内容を受領して処理を終了する（ステップＳ３８）。電子メールサーバ装置との接続に成功し、電子メールサーバ装置からのサービスの提供が可能になった場合、電子メールサーバ装置と対電子メールサーバ装置メッセージ交換手段7は、サービス受益対象が電子メールサービスが不要となるまで、その接続を継続する。もし、何らかの原因で、接続中の電子

7

メールサーバ装置がサービスを提供できなくなった場合は、次の障害発生時のフローにしたがって、サービスの再要求が行われる。

【0036】対電子メールサーバ装置メッセージ交換手段7が、電子メールサーバ装置に対してサービス要求を送信した後、あらかじめ定めた時間、サービス内容が返送されない場合、対電子メールサーバ装置メッセージ交換手段7は、サービス内容の受信に失敗したものと判断し（ステップS41）、電子メールサーバ装置に対してサービスの再要求を行う。すなわち、ステップS13－S23において、電子メールサーバ装置を選択し、選択した電子メールサーバ装置に対して、サービス再要求を送信する（ステップS32）。なお、この手順は、図2の接続時の手順と同様である。この手順は、サービス受益対象に何ら通知せずに行われる。すなわち、サービスを提供する電子メールサーバ装置が障害を起こすたびに、対電子メールサーバ装置メッセージ交換手段7は、他のサービス可能な電子メールサーバ装置とクライアント装置とを順次接続し、サービスの要求およびサービス内容の転送を実行する（ステップS32－S38）。こうして、無停止の電子メールシステムを実現することができる。

【0037】ただし、サービス可能な電子メールサーバ装置が無くなった場合には、対電子メールサーバ装置メッセージ交換手段7は、サービス受益対象に対して、サービス提供不能通知を送信し（ステップS15）、サービス受益対象は、電子メールサーバ装置が障害である旨の報告を行う（ステップS16a）。

【0038】

8

【発明の効果】以上説明したように、この発明は、電子メールサーバ装置の切り替えをクライアント装置側で行うことによって、安価な電子メールシステムを実現することができる。また、処理負荷の最も軽い電子メールサーバ装置を選択する方式を使用することによって、負荷分散を実現することができる。

【図面の簡単な説明】

【図1】この発明による電子メール多重化システムの実施例の構成を示すブロック図である。

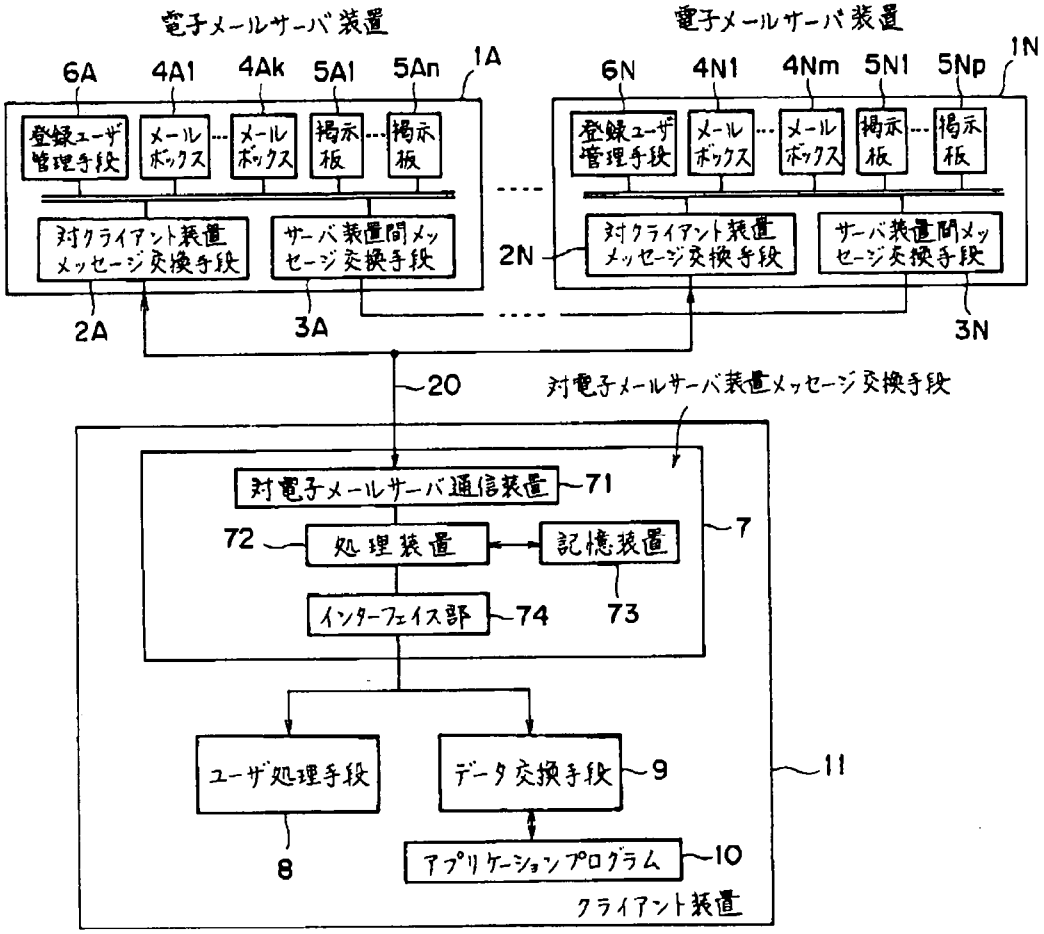
【図2】上記実施例の接続時の動作を示すフローチャートである。

【図3】上記実施例のサービス要求時の動作を、正常時と、障害発生時とに分けて示すフローチャートである。

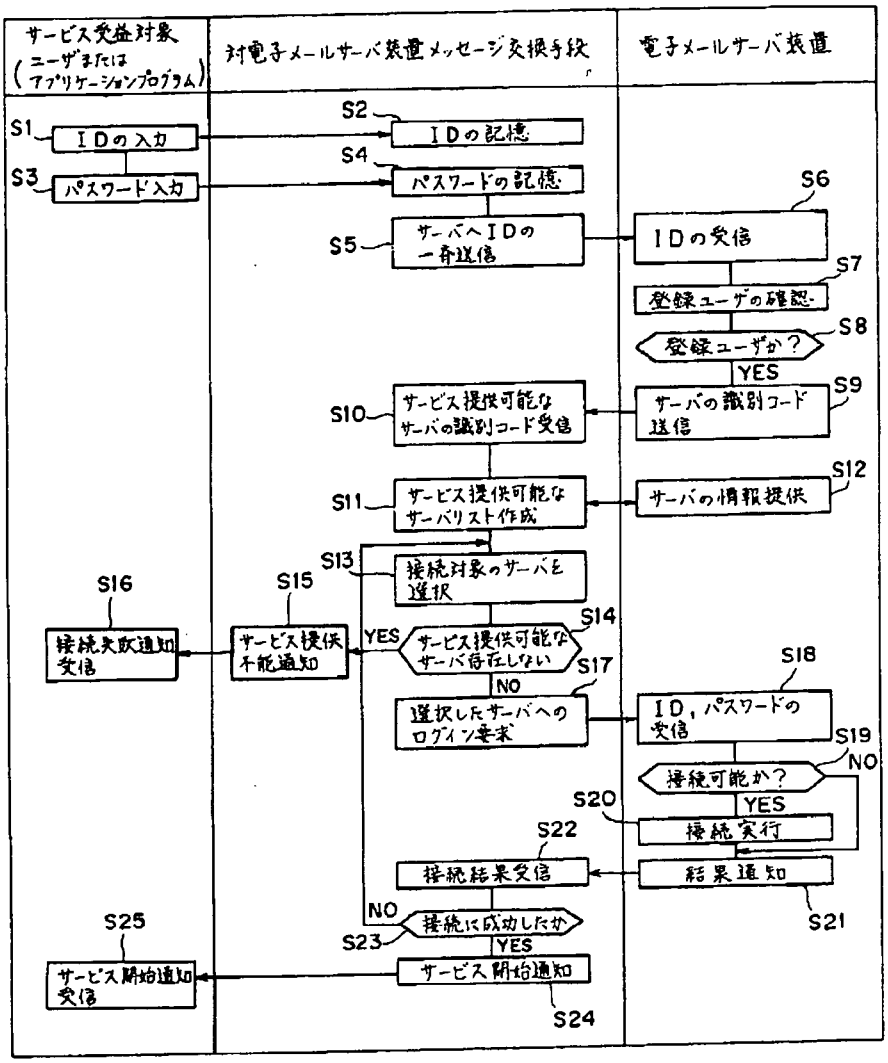
【符号の説明】

1A，1N　電子メールサーバ装置
2A，2N　対クライアント装置メッセージ交換手段
3A，3N　サーバ装置間メッセージ交換手段
4A1－4Ak，4N1－4Nm　メールボックス
5A1－5An，5N1－5Np　掲示板
6A－6N　登録ユーザ管理手段
7　対電子メールサーバ装置メッセージ交換手段
8　ユーザ処理手段
9　データ交換手段
10　アプリケーションプログラム
20　ネットワーク
71　対電子メールサーバ通信装置
72　処理装置
73　記憶装置
74　インターフェイス部

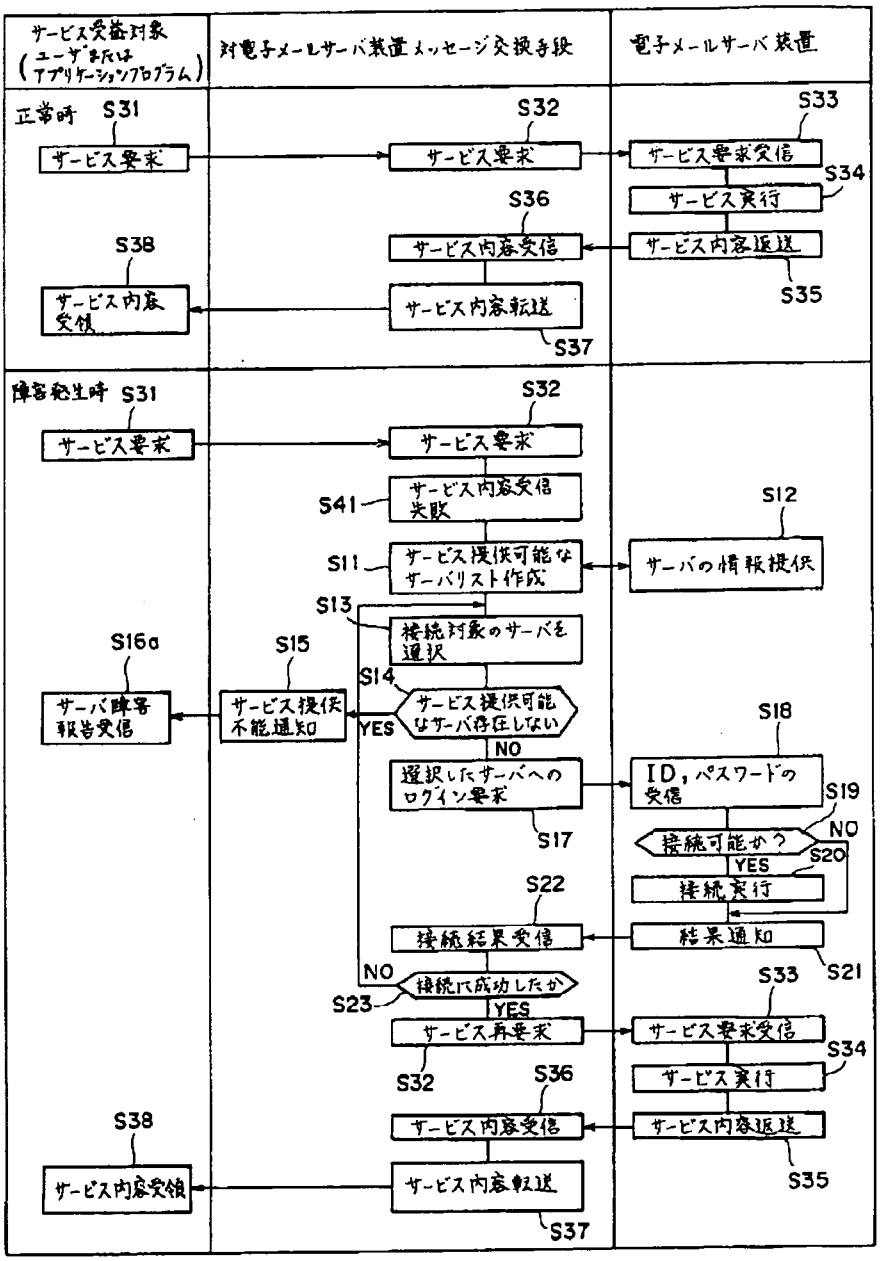
【図1】


電子メールサーバ装置
電子メールサーバ装置
6A
4A1
4Ak
5A1
5An
1A
6N
4N1
4Nm
5N1
5Np
1N
登録ユーザ管理手段
メールボックス
メールボックス
掲示板
掲示板
登録ユーザ管理手段
メールボックス
メールボックス
掲示板
掲示板
対クライアント装置メッセージ交換手段
サーバ装置間メッセージ交換手段
2N
対クライアント装置メッセージ交換手段
サーバ装置間メッセージ交換手段
2A
3A
3N
20
対電子メールサーバ装置メッセージ交換手段
対電子メールサーバ通信装置
71
72
処理装置
記憶装置
7
インターフェイス部
74
73
ユーザ処理手段
データ交換手段
9
11
8
アプリケーションプログラム
10
クライアント装置

【図2】


サービス受益対象
(ユーザまたは
アプリケーションプログラム)
対電子メールサーバ装置メッセージ交換手段
電子メールサーバ装置
S1
IDの入力
S2
IDの記憶
S3
パスワード入力
S4
パスワードの記憶
S5
サーバへIDの
一斉送信
S6
IDの受信
S7
登録ユーザの確認
S8
登録ユーザか？
YES
S9
サーバの識別コード
送信
S10
サービス提供可能な
サーバの識別コード受信
S11
サービス提供可能な
サーバリスト作成
S12
サーバの情報提供
S13
接続対象のサーバを
選択
S14
サービス提供可能な
サーバ存在しない
YES
S15
サービス提供
不能通知
S16
接続失敗通知
受信
NO
S17
選択したサーバへの
ログイン要求
S18
ID，パスワードの
受信
S19
接続可能か？
NO
YES
S20
接続実行
S21
結果通知
S22
接続結果受信
NO
S23
接続に成功したか
YES
S24
サービス開始通知
S25
サービス開始通知
受信

【図3】


サービス受益対象（ユーザまたはアプリケーションプログラム）
対電子メールサーバ装置メッセージ交換手段
電子メールサーバ装置
正常時
S31
サービス要求
S32
サービス要求
S33
サービス要求受信
S34
サービス実行
S36
サービス内容受信
サービス内容返送
S35
S38
サービス内容受領
サービス内容転送
S37
障害発生時
S31
サービス要求
S32
サービス要求
S41
サービス内容受信失敗
S12
S11
サービス提供可能なサーバリスト作成
サーバの情報提供
S13
接続対象のサーバを選択
S16a
S15
S14
サーバ障害報告受信
サービス提供不能通知
YES
サービス提供可能なサーバ存在しない
NO
S18
遅択したサーバへのログイン要求
ID，パスワードの受信
S19
S17
接続可能か？
NO
YES
S20
接続実行
S22
接続結果受信
結果通知
S21
NO
接続に成功したか
S33
S23
YES
サービス再要求
サービス要求受信
S34
S32
S36
サービス実行
S38
サービス内容受信
サービス内容返送
S35
サービス内容受領
サービス内容転送
S37

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 801 487 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
15.10.1997 Bulletin 1997/42

(51) Int. Cl.[6]: **H04L 29/06**, H04L 12/00

(21) Application number: **97105946.4**

(22) Date of filing: **10.04.1997**

(84) Designated Contracting States:
DE FR GB

(30) Priority: **10.04.1996 US 631720**

(71) Applicant: **AT&T Corp.**
New York, NY 10013-2412 (US)

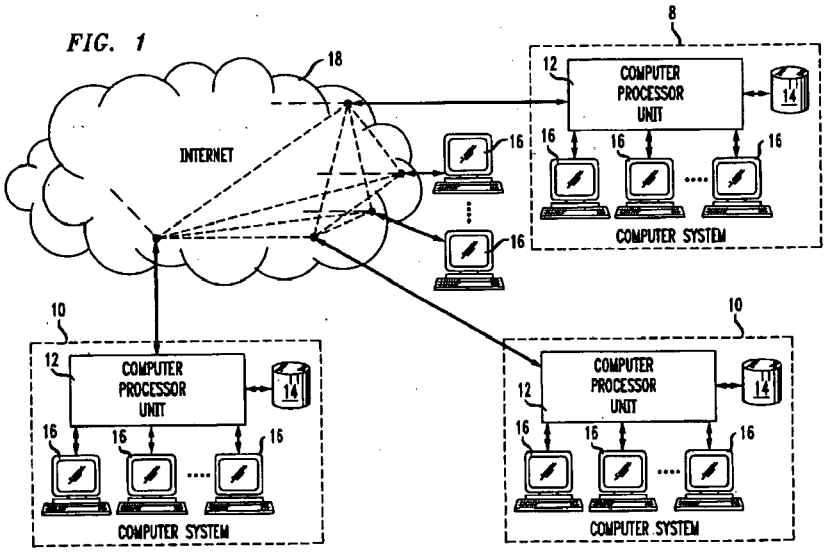
(72) Inventor: **Douglas Blewett Charles**
Madison ,N.J 07940 (US)

(74) Representative: **KUHNEN, WACKER & PARTNER**
Alois-Steinecker-Strasse 22
85354 Freising (DE)

(54) A url rewriting pseudo proxy server

(57) A method for real time remapping of access to a selected remote domain in an interconnected computer system network comprising the steps defining a pseudo proxy server and translating in the pseudo proxy server a remote record identifier corresponding to the remote domain to a remapped record identifier corresponding to the local domain. In a further enhancement the method comprises the additional step of determining if a selected record identifier is a selected remapped record identifier.

FIG. 1



EP 0 801 487 A2

Printed by Rank Xerox (UK) Business Services
2.14.16/3.4

Description

FIELD OF THE INVENTION

This invention relates to the field of interconnected computers, and more particularly to the field of formatted data distributed on interconnected computers.

BACKGROUND OF THE INVENTION

Because the Internet evolved from the ARPAnet, a research experiment that supported the exchange of data between government contractors and (often academic) researchers, an on-line culture developed that is alien to the corporate business world. The Internet was not designed to make commercialization easy.

Domain names direct where e-mail is sent, files are found, and computer resources are located. They are used when accessing information on the WWW or connecting to other computers through Telenet. Internet users enter the domain name, which is automatically converted to the Internet Protocol address by the Domain Name System (DNS). The DNS is a service provided by TCP/IP that translates the symbolic name into an IP address by looking up the domain name in a database.

The World Wide Web (WWW) is one of the newest Internet services. The WWW allows a user to access a universe of information which combines text, audio, graphics and animation within a hypermedia document. Links are contained within a WWW document which allows simple and rapid access to related documents. The WWW was developed to provide researchers with a system that would enable them to quickly access all types of information with a common interface, removing the necessity to execute a variety of numerous steps to access the information. During 1991, the WWW was released for general usage with access to hypertext and UseNet news articles. Interfaces to WAIS, anonymous FTP, Telnet and Gopher were added. By the end of 1993 WWW browsers with easy to use interfaces had been developed for many different computer systems.

With HyperText Markup Language (HTML) based pages, such as the WWW, the pages of information contain pointers to other pages. The pointers, are links which are encoded with Uniform Resource Locators (URLs). The URL contains a transmission protocol, such as HyperText Transfer Protocol (HTTP), a domain name of the target computer system, and a page identifier.

Accordingly, with the commercialization of the Internet through advertising, charging for access to information, and other schemes there is a need for an Internet Service Provider (ISP) to record all of the interactions that their customers have with HTML based content.

SUMMARY OF THE INVENTION

In an interconnected computer system network there is provided a method for real time remapping of a remote domain to a local domain. The method comprising the steps defining a pseudo proxy server and translating in the pseudo proxy server a remote record identifier corresponding to the remote domain to a remapped record identifier corresponding to the local domain. In a further enhancement the method comprises the additional step of determining if a selected record identifier is a selected remapped record identifier.

In an enhancement of the present invention, there is provided a method of providing pseudo proxy access for tracking and controlling access to remote record identifiers. The method comprising the steps of: providing a first data set having rewritten record identifiers for a remote record identifier to a local user; responding to a request from the local user for a selected record identifier; determining if the selected record identifier is a rewritten record identifier; determining an actual record identifier for the rewritten record identifier; and requesting a second data set corresponding to the actual record identifier from said interconnected computer system network.

In an another enhancement of the present invention the first data set comprises a HyperText Markup Language based data set.

In a further enhancement of the present invention the remote record identifier comprises a uniform record locator.

In yet a further enhancement the present invention comprises the additional steps of determining that a record identifier is remote and rewriting the remote record identifier.

Determining that the record identifier is remote in an enhancement of the present invention comprises the step of scanning a domain name of the actual record identifier and comparing the domain name to a local domain name wherein the record identifier is remote if the domain name is different than the local domain name.

In yet further enhancements of the present invention the step of determining an actual record identifier for the rewritten record identifier comprises looking up the actual record identifier by a predetermined index, by a hashing table, by addressing a memory location, by accessing an inode of a disk file, or by accessing a disk file by a file name.

BRIEF DESCRIPTION OF THE DRAWINGS

A more complete understanding of the present invention may be obtained from consideration of the following description in conjunction with the drawings in which:

FIG. 1 is an overview of interconnected computer system networks employing the present invention; and

FIG. 2 is a flow chart of the procedures of the

present invention which for tracking local access and local control by rewriting URL's.

DETAILED DESCRIPTION OF VARIOUS ILLUSTRATIVE EMBODIMENTS

Although the present invention is particularly well suited for use as a URL rewriting pseudo proxy server for the WWW, and shall be described with respect to this application, the methods and apparatus disclosed here can be applied to other schemes employing URLs as well as other types of resource location pointers and other record identifiers as links within an interconnected computer system network.

The WWW allows a user to access a universe of distributed information which combines text, audio, graphics and animation within a hypermedia document. Links are contained within a WWW document which allows simple and rapid access to related documents. The WWW provides an access system that enables users to quickly access all types of information with a common interface, removing the necessity to execute a variety of numerous steps to access the information. The WWW supports interfaces for access to HyperText, UseNet news, WAIS, anonymous FTP, Telnet and Gopher.

The WWW has HTML based pages, which contain pointers to other pages. The pointers, are HyperText links which are encoded with URLs. The URL contains a transmission protocol, such as HTTP, a domain name of the target computer system, and a page identifier.

The HyperText links are simply references to other documents, made up of two parts. The first part is a reference to a related item such as a document, picture, movie or sound. The item being referenced can be within the current document, or it can be located anywhere on the Internet. The second part is an anchor. The anchor can be defined to be a word, group of words, a picture, or any area of the display. A reader activates an anchor by pointing to it and clicking with a mouse, when using a graphical browser, or by selecting it with the cursor (arrow) keys or tab keys, when using a texted based browser. Anchors can be indicated in the displayed document by color, graphics, reverse video, underline as well as other formats.

When an anchor is activated, the browser fetches the item referenced by the anchor. This may involve reading a document from a local disk drive, or requesting over the Internet that a document be sent to the local computer.

The standard way an item is referenced is by a URL. The URL contains a complete description of the item, which is made up of a protocol and an address. An absolute address reference contains the complete address including domain name, directory path, and file name. A relative address reference assumes that the previous domain name and directory path are used.

The URL is not limited to identifying WWW HyperText files, but can also access other sets of data in different protocols including anonymous FTP, Gopher, WAIS, UseNet news, and Telenet. The URL format is typically *P://A*. *P* is the protocol, such as HTTP (HyperText Transfer Protocol), gopher, FTP (file transfer protocol), WAIS (Wide Area Information Server), news (UseNet newsgroups), or Telenet. *A* is a valid Internet host address or symbolic location.

To better understand the present invention, an example of an embodiment in which a newspaper consortium composed of individual members are interconnected through the Internet shall be used. An individual member may want to provide access to all of the consortium member organizations, but would only track their local subscribers.

Referring to FIG. 1 there is shown an overview of interconnected computer system networks. Each computer system network 8 and 10 contains a local computer processor unit 12 which is coupled to a local data storage unit 14. The local computer processor unit 12 is selectively coupled to a plurality of local users 16. Each of the computer processor units 12 are selectively coupled to other computer processor units 12 through the Internet 18. Local users 16 are also selectively coupled directly to the Internet 18.

A local newspaper, which has a computer system network 8, such as the Local Paper in Wyoming, may allow a local user 16 to click into another computer system network 10, such as a Regional Paper in New York through the Internet 18 to access a data storage unit 14 on the other computer system network 10. The Local Paper computer system network 8 would handle the billing for the local user 16 and provide an authentication and reconciliation scheme with the Regional Paper computer system network 10, permitting both papers to profit from the venture.

The current technology utilized over the Internet, specifically HTML based pages does not provide a suitable means for achieving the desired scheme. The HTML based pages contain hyper links encoded as URLs, to other pages. If we assume that the Regional Paper has a machine (domain) name of regional-paper.com for its computer system network 10 and a HTML page about regional news today called regional-today.html, a URL pointing to the regional news today at the Regional Paper would be

http://regional-paper.com/regional-today.html

which allow access to the appropriate HTTP page through the Internet 18. In this case the URL acts as a remote record identifier.

If this URL is included in an HTML page served by the computer system network 8 of the Local Paper HTTP server, the computer system network 8 of the Local Paper would have no way of telling if or when the local user 16 accessed the regional-today page on the other computer system network 10. Selecting the URL results in the other computer system network 10 of the regional-paper.com being accessed and the computer system network 8 of the local-paper.com is not involved in the access.

An eloquent way to achieve tracking of access and local control is to make all of the URL's local to the local-paper.com machine, thus permitting the local-paper.com machine to track and control access. Referring to FIG. 2, there is shown a flow chart of the procedures of the present invention which accomplishes tracking of local access and local control by rewriting the URL's on the fly as they pass through the local-paper.com machine in being served to the local user. In step 20, it is determined on the fly if the actual URL is remote and if it is to be rewritten. All remote URL's may be selected for rewriting, or selective groups may be selected for rewriting. The selection may be based upon the remote domain name which can be compared with a list of remote domain names that are to be tracked as well as other comparison criteria. Thus, in step 22 the selected remote URL

http://regional-paper.com/regional-today.html

would be rewritten as

http://local-paper.com/127.html

while the text and graphics on the HTML page would remain the same.

In step 24 the local system sends the HTML page containing the rewritten URLs to the local user. In step 26 the local user clicks on (selects) a URL on the HTML page, thus requesting the document 127.html from the local-paper.com machine. In step 28 the local HTTP server determines if this is a rewritten URL. If the URL is rewritten, step 30 looks up the actual URL, and in step 32 sends the HTML page from the regional-paper.com machine. If the URL was not rewritten, step 30 is skipped. It is highly desirable that the rewritten URL's be "blind" and not easily decoded, in order that a user could not easily defeat the rewriting mechanism. After step 30 the procedure can repeat again from step 20.

The local-paper.com machine by serving up the HTML page from the regional-paper.com is acting as a pseudo proxy. Proxy servers are often employed in environments that contain firewalls. There, the proxy acts on behalf of the user through the firewall, directing all HTTP access through it is not desirable to supply proxy service to every user. Many users access the Internet through a corporate firewall. It is desirable to leave the user's environment(s) unchanged. The URL rewrite scheme does this by being completely transparent to the end user. URL's that are not rewritten, which are links that we do not want to track, are not rewritten and behave as usual.

The proxy server in the rewrite scheme is a pseudo proxy or domain specific proxy, in that the server only acts as a proxy for the HTML pages that it is hosting and the pages that it is pointing to. Typically, proxies have all or no requests sent through them. In the present invention, with the pseudo proxy, only the requests in its domain are served. The conversion of the original remote URL to a local/pseudo proxy based URL can be implemented efficiently. The rewriting of the URLs is a remapping of selected record identifies from one domain to another domain (between a local and a remote domain).

First the URL is recognized as a remote URL, which is shown as step 20. This can be accomplished by scanning the domain name part of the URL. If the domain name is remote when compared to a local domain name or when compared to a predetermined table of domain names that are to be tracked, the remote URL is replaced by an opaque local URL which is shown in step 22. An opaque URL is one that the user can not easily generate or reconstruct the remote URL from, as this would subvert the process. This can be accomplished by using indices that are private to the HTTP server. The generation of the indices can be accomplished from a local register, an incremented integer, or memory address from where the string is stored in a database, the inode of a disk file, or a simple disk file name.

The conversion of the proxy URL can be done by using indices. The number is an index into an array where the actual remote URL is stored, utilizing a minimal perfect hash. Hashing is a technique for arranging a set of items, in which a hash function is applied to the key of each item to determine its hash value. The hash value identifies each item's primary position in a hash table, and if this position is already occupied, the item is inserted either into an overflow table or in another available position in the table.

The indices also provide a simple way of tracking access to the remote URLs, with the level of detail tracking limited only by the level of detail that is recorded. Further, the indices can be utilized to determined if access to the remote URL is to be granted or denied and may depend upon the particular status or identity of a local user.

An alternative scheme is when the name is a number of a memory address or a key stored in a database. Another alternative scheme is to utilize the disk inode which requires that the inode be looked up in the disk node table. When a disk file name is used, the file is opened which can contain the remote URL.

Numerous modifications and alternative embodiments of the invention will be apparent to those skilled in the art in view of the foregoing description. Accordingly, this description is to be construed as illustrative only and is for the purpose of teaching those skilled in the art the best mode of carrying out the invention. Details of the structure may be varied substantially without departing from the spirit of the invention and the exclusive use of all modifications which come within the scope of the appended claim is reserved.

Claims

1. In an interconnected computer system network a method for real time remapping a remote domain to a local domain, said method comprising the steps of:

   defining a pseudo proxy server; and

translating in said psuedo proxy server a remote record identifier corresponding to said remote domain to a remapped record identifier corresponding to said local domain.

2. The method as recited in claim 1 comprising the additional step of determining if a selected record identifier is a selected remapped record identifier.

3. The method as recited in claim 2 comprising the additional step of requesting a data set corresponding to a selected remote record identifier corresponding to said selected remapped record identifier.

4. The method as recited in claim 1 comprising the additional step of tracking access of a local user to said remote domain through said pseudo proxy server.

5. The method as recited in claim 1 comprising the additional step of restricting access of a local user to said remote domain through said pseudo proxy server.

6. The method as recited in claim 1 wherein said remote record identifier comprises a uniform record locator.

7. The method as recited in claim 1 wherein the step of translating a remote record identifier comprises looking up said remote record identifier by a predetermined index.

8. The method as recited in claim 7 wherein the step of translating a remote record identifier further comprises a hashing table.

9. The method as recited in claim 8 wherein said hashing table comprises a minimal perfect hash.

10. The method as recited in claim 7 wherein the step of translating a remote record identifier further comprises addressing a memory location.

11. The method as recited in claim 7 wherein the step of translating a remote record identifier further comprises accessing an inode of a disk file.

12. The method as recited in claim 1 wherein the step of translating a remote record identifier comprises accessing a disk file by a file name.

13. The method as recited in claim 3 wherein said data set comprises a hypertext transfer protocol request.

14. In an interconnected computer system network a method of tracking and controlling access to remote record identifiers, said method comprising the steps of:

providing a first data set having a rewritten record identifier for a remote record identifier to a local user;
responding to a request from said local user for a selected record identifier;
determining if said selected record identifier is a rewritten record identifier;
determining an actual record identifier for said rewritten record identifier; and
requesting a second data set corresponding to said actual record identifier from said interconnected computer system network.

15. The method as recited in claim 14 wherein said first data set comprises a hypertext markup language based data set.

16. The method as recited in claim 14 wherein said remote record identifier comprises a uniform record locator.

17. The method as recited in claim 14 comprising the additional steps of determining that a record identifier is remote and rewriting said remote record identifier.

18. The method as recited in claim 17 wherein the steps of determining that a record identifier is remote comprises scanning a domain name of said actual record identifier and comparing said domain name to a local domain name wherein said record identifier is remote if said domain name is different than said local domain name.

19. The method as recited in claim 14 wherein the step of determining an actual record identifier for said rewritten record identifier comprises looking up said actual record identifier by a predetermined index.

20. The method as recited in claim 19 wherein the step of determining an actual record identifier further comprises a hashing table.

21. The method as recited in claim 20 wherein said hashing table comprises a minimal perfect hash.

22. The method as recited in claim 19 wherein the step of determining an actual record identifier further comprises addressing a memory location.

23. The method as recited in claim 19 wherein the step of determining an actual record identifier further comprises accessing an inode of a disk file.

24. The method as recited in claim 14 wherein the step of determining an actual record identifier for said rewritten record identifier comprises accessing a

disk file by a file name.

25. The method as recited in claim 24 wherein said disk file contains a domain name of said actual record identifier.

26. The method as recited in claim 24 wherein said disk file contains said actual record identifier.

27. The method as recited in claim 14 wherein said request from said local user comprises a hypertext transfer protocol request.

28. In an interconnected computer system network a method of providing pseudo proxy access for tracking and controlling access to remote uniform record locators, said method comprising the steps of:

providing a hypertext markup language based page having a rewritten uniform record locator for a remote uniform record locator to a local user;
responding to a request from said local user for a selected uniform record locator;
determining if said selected uniform record locator is a rewritten uniform record locator;
determining an actual uniform record locator for said rewritten uniform record locator; and
requesting a second data set corresponding to said actual uniform record locator from said interconnected computer system network.

29. The method as recited in claim 28 comprising the additional steps of determining that a uniform record locator is remote by comparing a domain name of said uniform record locator to a local domain name, wherein said uniform record locator is remote if said domain name is different than said local domain name and rewriting said remote uniform record locator.

30. The method as recited in claim 28 wherein the step of determining an actual uniform record locator for said rewritten uniform record locator comprises looking up said actual uniform record locator by a predetermined index.

31. The method as recited in claim 30 wherein the step of determining an actual uniform record locator further comprises a hashing table.

32. The method as recited in claim 31 wherein said hashing table comprises a minimal perfect hash.

33. The method as recited in claim 28 wherein the step of determining an actual uniform record locator further comprises addressing a memory location.

34. The method as recited in claim 28 wherein the step of determining an actual uniform record locator further comprises accessing an inode of a disk file.

35. The method as recited in claim 28 wherein the step of determining an actual uniform record locator for said rewritten uniform record locator comprises accessing a disk file by a file name.

36. The method as recited in claim 28 comprising the additional steps of determining that a uniform record locator is remote by comparing a domain name of said uniform record locator to compared to a predetermined table of domain names and rewriting said remote uniform record locator.

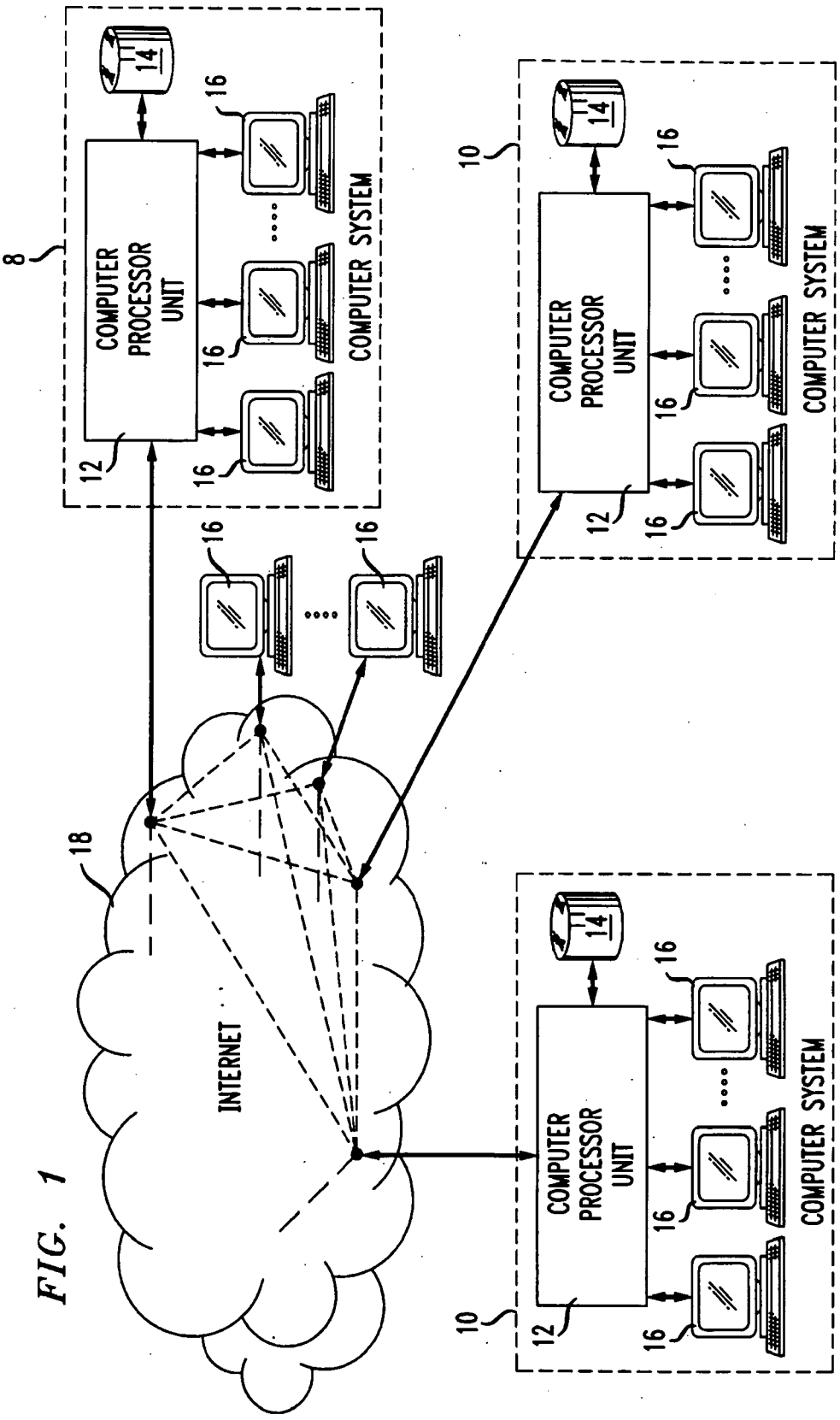
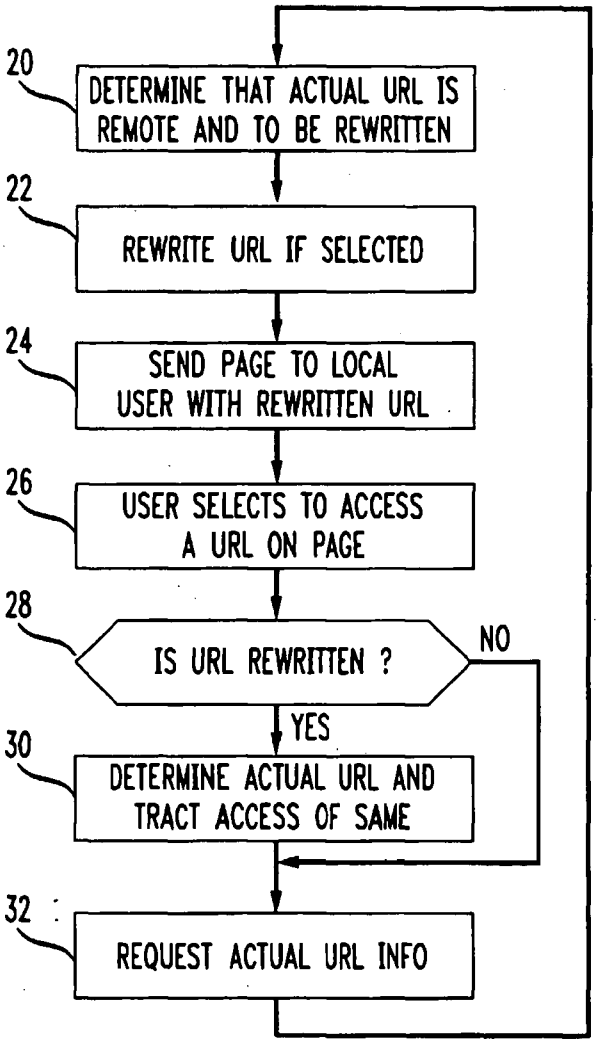

8
COMPUTER PROCESSOR UNIT
COMPUTER SYSTEM
12
14
16
10
18
INTERNET

FIG. 1

FIG. 2


20
DETERMINE THAT ACTUAL URL IS REMOTE AND TO BE REWRITTEN
22
REWRITE URL IF SELECTED
24
SEND PAGE TO LOCAL USER WITH REWRITTEN URL
26
USER SELECTS TO ACCESS A URL ON PAGE
28
IS URL REWRITTEN ?
NO
YES
30
DETERMINE ACTUAL URL AND TRACT ACCESS OF SAME
32
REQUEST ACTUAL URL INFO

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 824 236 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
18.02.1998 Bulletin 1998/08

(51) Int Cl.[6]: **G06F 9/46**

(21) Application number: **97304739.2**

(22) Date of filing: **01.07.1997**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV RO SI

(30) Priority: **08.08.1996 GB 9616649**

(71) Applicant: **INTERNATIONAL BUSINESS MACHINES CORPORATION**
Armonk, NY 10504 (US)

(72) Inventors:
- **Baird, Richard**
  Markham, Ontario L3R 8V3 (CA)
- **Ogilvie, Alan**
  Southampton, Hampshire SO52 9NA (GB)

(74) Representative: **Davies, Simon Robert**
I B M
UK Intellectual Property Department
Hursley Park
Winchester, Hampshire SO21 2JN (GB)

(54) Computer system

(57) A computer workstation 1 is attached to a network such as the Internet 34. Browser software 210 running on the computer workstation can access and retrieve pages of data from over the network, with the pages of data then being displayed at the computer workstation. A page of data can include a reference to code which the browser can also retrieve from over the network and then execute on the computer workstation. The reference to the code also has associated with it a reference to another file, again accessible over the network, which contains information pertaining to the code, such as the size of the code. The browser accesses the information in this file, prior to retrieving or executing the code. This information can then help the user at the computer workstation determine whether or not to retrieve the code.

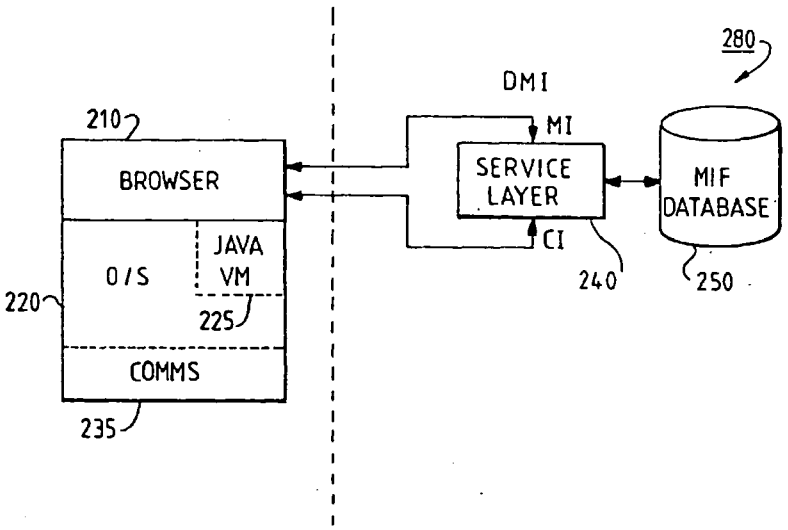


FIG. 2

EP 0 824 236 A2

Printed by Jouve, 75001 PARIS (FR)

Description

The present invention relates to a computer system, and to a method of operating such a computer system attached to a network, such that the computer system can access data and code over the network.

In the past couple of years there has been an explosive growth in the Internet, and in particular of the WorldWide Web (WWW), which is one of the facilities provided on top of the Internet. The WWW comprises many pages or files of information, distributed across many different servers. Each page is identified by a Universal Resource Locator (URL). The URL denotes both the server machine, and the particular file or page on that machine. There may be many pages or URLs resident on a single server.

In order to use the WWW, a client runs a piece of software known as a Web browser, such as WebExplorer (provided as part of the OS/2 operating system from IBM Corporation), or the Navigator program available from Netscape Communications Corporation. The client interacts with the browser to select a particular URL, which in turn causes the browser to send a request for that URL or page to the server identified in the URL. Typically the server responds to the request by retrieving the requested page, and transmitting the data for that page back to the requesting client (the client server interaction is performed in accordance with the hypertext transport protocol ("http")). This page is then displayed to the user on the client screen. The client may also cause the server to launch an application, for example to search for WWW pages relating to particular topics.

Most WWW pages are formatted in accordance with a language known as html (hypertext mark-up language). Thus a typical page includes text together with embedded formatting commands, referred to as tags, which can be used to control the font size, the font style (for example, whether italic or bold), how to lay-out the text, and so on. A Web browser parses the html script in order to display the text in accordance with the specified format. In addition, an html page can also contain a reference, in terms of another URL, to a piece of multimedia data, for example, an image, a video segment, or an audio file. A Web browser responds to such a reference by retrieving and displaying or playing the data. Alternatively, such multimedia data may form its own WWW page, without any surrounding html text.

Most WWW pages also contain one or more references to other WWW pages, which need not be on the same server as the original page. Such references may generally be activated by the user selecting particular locations on the screen, typically by (double) clicking a mouse control button. These references or locations are known as hyperlinks, and are typically flagged by the browser in a particular manner (for example, any text associated with a hyperlink may be in a different colour). If a user selects the hyperlink, then the referenced page is retrieved and replaces the currently displayed page.

Further information about html and the WWW can be found in "World Wide Web and HTML" by Douglas McArthur, p18-26 in Dr Dobbs Journal, December 1994, and in "The HTML SourceBook" by Ian Graham, (John Wiley, New York, 1995).

As so far described, and broadly speaking as currently implemented, the WWW suffers from the disadvantage that pages downloaded from a server to a client are essentially passive, in other words, they do not contain code which is executed at the client machine. One implication of this is that the server cannot offload onto the client any of the processing associated with the interaction between the client and the server. Thus if the client is completing a form with their telephone number for example, then any formal checks such as to the number of digits in the telephone number must be performed at the server. This results firstly in a heavier processing burden at the server, and secondly in time-consuming extra communications between the server and client should there be any mistakes to correct. Moreover, the inability of the server to download code for execution at the client is a significant limitation on the type of applications that can be created to exploit the WWW.

Recent developments, based particularly on the Java technology from Sun Microsystems Inc, have sought to overcome the above difficulties. The Java technology comprises primarily (i) a new programming language, somewhat similar to C and C++, and (ii) a virtual machine. Essentially, programs written in the Java programming language can be compiled into byte code form, and then interpreted at runtime on the Java virtual machine. The Java virtual machine converts the byte codes into instructions that can be executed by the underlying physical machine.

Programs written using Java can be downloaded over the WWW in the form of byte codes for execution on a Java virtual machine at the client. Such programs are known as "applets". The use of the Java technology for downloading code over the WWW has two major benefits. Firstly, an applet can be platform independent, if we assume that each client has a copy of the Java virtual machine (the virtual machine at the client's system is typically incorporated either into the operating system, or into the Web browser itself). In other words, there is no need for a server to have different versions of the code for downloading to clients according to their respective operating systems and machines. Therefore, only a single version of the relevant code needs to be written and maintained, which makes life much simpler for software developers. Secondly, because the applet executes on a virtual machine, rather than a physical machine, security is greatly improved. Thus, when downloading code over the network, there is always a risk that it will include some malicious code (accidentally or otherwise) that may damage data or programs stored at the client. The virtual machine however can monitor the operation of the applet, and so detect and prevent such malicious activity.

It will be noted that the concept of downloading software from a server to a client in the form of byte codes for execution on a virtual machine was also known independently of the Java technology, see for example US 5347632.

In order to invoke a Java applet, a Web page of html text contains an <APPLET> tag, which identifies the URL containing the applet. A browser responds to this tag by retrieving and running the applet. Also defined is a <PARAM> tag, which is contained within a pair of corresponding <APPLET> and </APPLET> tags, and which can be used to specify parameters that are passed to the applet at run-time. (Note that the APPLET and PARAM tags are not formally incorporated into the html standard, but are nevertheless recognised by many Web browsers). Further information about the Java technology and applets can be found in "Teach Yourself Java in 21 Days" by Laura Lemay and Charles Perkins (Sams.net Publishing, Indianapolis, USA, 1996).

One of the difficulties with the above approach is that the amount of information that can be contained within the APPLET tag is limited. This can lead to problems if the browser has to decide whether or not to download a particular applet referenced within an html page. For example, the browser may not wish to download an applet which is particularly large (and hence slow to transport across the network), or which requires a more recent version of the Java virtual machine than the one currently installed at the client.

Accordingly, the invention provides a method of operating a computer attached to a network, having accessible at least one page of data that includes a reference to code that can be retrieved over the network for execution at said computer, characterised by the steps of: locating, associated with said reference, an identifier to information about said code; and retrieving the information about the code.

Thus the computer can utilise the identifier to locate and retrieve information concerning the code. This information may then be used in order to determine whether or not to run said code. For example, the information preferably states the size of the code, and it may be decided that the code is too large to be downloaded within a reasonable time; another possibility is that the information specifies that execution of the code requires hardware support features, such as a video capture facility, which may not be installed at the client workstation. Such a determination may be explicitly surfaced to the user, or may be performed automatically based on predetermined criteria (or a combination of both automatic and user-prompted decision making could be adopted).

Other possible uses for the information about the code might include for audit purposes, perhaps in circumstances where there is some royalty payment attached to use of the code, or for simple informational purposes, such as providing access to a copyright notice and licensing terms, and so on. Another alternative is to use the information to assist in invoking the code itself. For example, if the code is encrypted, then the information may specify the algorithm needed to decrypt the code before it can be invoked.

In the preferred embodiment, once the information about the code has been retrieved and stored and/or processed as desired, the code itself is then retrieved over the network. Alternatively, the code might be retrieved prior to retrieval of the information about the code.

In the preferred embodiment, the at least one page of data is in html format, the reference comprises an APPLET tag, and the identifier comprises one or more attributes within said APPLET tag. Preferably said reference and said identifiers are in the form of Universal Resource Locators (URLs), with each reference and identifier being specified in two parts, the first specifying server and directory, the second the actual file name. In this preferred embodiment, the applet is downloaded in Java byte code form, for execution on a Java virtual machine at the client's workstation. This represents the typical situation where the computer is running a Web browser, and said at least one page of data, said code reference and said information are all accessible over the Internet.

It will be appreciated that since the specific APPLET tag is particularly associated with the Java language, it may be replaced in some future standard with a more generalised tag (ie one that is regarded as more language-neutral), although of course this would not affect the principle of the present invention. It will also be appreciated that in some circumstances, said at least one page of data may contain little or nothing apart from the reference to the code, and its associated identifier of the information (in other words, the primary or sole purpose of the page may be to access the code).

In a preferred embodiment, the information conforms to the Management Information Format (MIF). This is a well-known standard, and so is likely to be understood by the client trying to access the code. However, the information could be stored in any format which the browser can understand and access. The MIF information is copied over the network and then preferably loaded into the MIF database at the computer, where it can be interrogated by the browser (or simply stored for audit purposes). Alternatively, the MIF information may already be stored in a remote MIF database, so that information about the code can be retrieved by remotely interrogating said database over the network. This saves having to transmit the whole MIF file over the network.

Thus the invention provides a mechanism for making available information about code which can be accessed over the network. An alternative to the approach provided by the invention might be for example to have information relating to code present in an html page itself, for example as additional attributes in the <APPLET> tag. However, this would not only lead to the tags becoming potentially very large, but also each page which referenced a particular applet would then have to contain a copy of this information. This is not only more expensive in terms of storage requirements,

but also extremely difficult to maintain from a practical point of view, since changes to the applet have to be reflected at multiple locations (indeed the owner of the code will generally be unaware of all the sites which reference that code). Another alternative might be for the information about code to be identified within the code itself, so that on execution the code initially brings up the relevant information. This strategy has the drawback that it requires the code to always be at least partly executed. Thus it can be seen that the approach adopted by the present invention represents the most powerful, flexible, and efficient way of making available information about code which can be accessed over the network.

The invention also provides a computer system, attachable to a network to access at least one page of data that includes a reference to code that can be retrieved over the network for execution at said computer, characterised by: means for locating, associated with said reference, an identifier to information about said code; and means for retrieving the information about the code.

The invention further provides a computer program product for execution by a computer system attached to a network to access at least one page of data that includes a reference to code that can be retrieved over the network for execution at said computer system, said computer program product including: means for locating, associated with said reference, an identifier to information about said code; and means for retrieving the information about the code.

The computer program product typically resides on a diskette or a CD-ROM, or alternatively may be stored on a tape or hard disk of a server machine for downloading across the network, and in use is copied into the working memory (typically RAM) of the computer system.

The invention further provides a Web browser for accessing at least one page of data on the World Wide Web that includes a reference to applet code that can be retrieved over the Internet for execution at said computer, characterised by: means for locating, associated with said reference, an identifier to information about said applet code; and means for retrieving the information about the code.

The invention further provides a method of operating a server attached to a network, said method including steps of: storing at least one page of data that includes a reference to code that can be retrieved over the network, and an identifier associated with said reference to indicate the location of information stored about said code; receiving a request over the network for said at least one page of data; and responsive to said request, transmitting said at least one page of data.

A preferred embodiment of the invention will now be described in detail by way of example only with reference to the following drawings:

Figure 1 is a simplified schematic diagram of a computer system;

Figure 2 is a simplified schematic diagram of the main software components on the computer system of Figure 1;

Figure 3 is a flowchart illustrating processing according to a first embodiment of the invention; and

Figure 4 is a flowchart illustrating processing according to a second embodiment of the invention.

Figure 1 is a simplified schematic diagram of a computer system 1, including a system unit 10, a display screen 12, a keyboard 14 and a mouse 16. The system unit includes a microprocessor 22, semi-conductor memory (ROM/RAM) 24, a hard disk drive 28, and a bus over which data is transferred 26. A network adapter card 30 is provided, which, in conjunction with link 32, enables the computer workstation to exchange data with the Internet 34. The computer is essentially controlled by the microprocessor, based on software instructions loaded into ROM/RAM, typically from the hard disk drive, or from the network. The system of Figure 1 may be any conventional workstation, such as a Personal Computer machine or a RISC System/6000 computer, both available from IBM Corporation. Such systems are very well-known in the art, and the skilled person is aware of great variety of possible system configurations: for example, the system may include a CD-ROM, have multiple network adapter cards, use a telephone modem instead of a network adapter card for the network connection, include audio systems, and so on. Such variations are not pertinent to an understanding of the present invention.

Figure 2 illustrates the software components running on system 1, including the operating system 220 and Web browser 210 (note that in some cases the browser may be incorporated into the operating system). As previously discussed, the browser is responsible for accessing material over the Internet, and displaying or otherwise processing the retrieved material as appropriate. One important feature of the browser is its ability to understand html commands. The operating system 220, such as AIX or OS/2, both available from IBM Corporation is responsible for general system management of the workstation. Included within the operating system is communication software 235 which allows the workstation to interact with the Internet 34. As is well known in the art, such communications are based on the TCP/IP protocol. The operating system also includes a Java virtual machine 225, which can interpret Java byte codes in order to run Java applications. Note that in some cases the Java virtual machine may be provided as part of the

browser. Also running in one embodiment on system 1 is Desktop Management Interface (DMI) software 280 (which also may be incorporated into the operating system 220). This will be described in more detail later.

System 1 and its software can be used to access and process WWW pages, including Java applets, in known fashion, using the http protocol. An example of the html text in a Web page which references a Java applet is as follows:

```
<APPLET CODEBASE = "http://ncc.hursley.ibm.com/javainfo/classes"
      CODE = "Map.class"
      WIDTH = 560
      HEIGHT = 560
>
<param name = IMAGE value = "graphics/cup-bun.jpg">
<param name = MAP value = "java.map">
<param name = TYPE value = "NCSA" >
</APPLET>
```

Example 1

The browser interprets this html text as follows. In response to the <APPLET> and </APPLET> tags, the browser retrieves the code which is located in file "Map.class" at the server and directory specified by "http://ncc.hursley.ibm.com/javainfo/classes". (As is well-known in html, tags of the form "</xxx>" mark the end of information or data associated with the preceding corresponding <xxx> tag). This processing is essentially the same as accessing any WWW page from a URL. The browser then creates an applet context object. Associated with this are the values of HEIGHT and WIDTH (known as attributes), which define the amount of screen which the applet needs for execution, and also certain parameter values. These parameter values are listed in their own <PARAM> tags which specify the name and value of the tag, and can be retrieved into the applet itself by the use of the Java getParameter method. Thus for example TYPE is the name of a string variable in the Map.class applet, which is to be assigned the value "NCSA". Finally, the applet is executed by creating an applet object using the parameter data previously stored in the applet context object and then invoking the applet object using the Java **init()** instruction.

As mentioned previously, some difficulties can arise with this known approach, for example, the Java applet may require a more recent version of the Java virtual machine than is installed at the client. In this situation, if the client tries to download and run the applet, he or she may discover that operation of the applet is limited or impossible.

The present invention overcomes this problem by expanding the information presented to the browser relating to the applet. Thus, in accordance with the present invention, the html text of Example 1 would be modified as follows:

```
<APPLET CODEBASE = "http://ncc.hursley.ibm.com/javainfo/classes"
      CODE = "Map.class"
      MIFBASE= "http://ncc.hursley.ibm.com/javainfo/mifs"
      MIF="Map.mif"
      WIDTH = 560
      HEIGHT = 560
>
<param name = IMAGE value = "graphics/cup-bun.jpg">
<param name = MAP value = "java.map">
<param name = TYPE value = "NCSA" >
</APPLET>
```

Example 2

It will be noted that Example 2 is the same as Example 1, except for the addition of two new attributes into the <APPLET> tag. The two attributes, MIFBASE and MIF, are analogous to the fields CODEBASE and CODE, in that they specify a URL comprising a server/directory and file respectively. However, whereas CODEBASE and CODE locate the actual code for the Java applet, MIFBASE and MIF locate a file containing information describing the applet. This file may reside on the same server and possibly in the same directory as the applet code itself, or may reside on a completely different server.

In the preferred embodiment, the file containing the information describing the applet is in a particular format known as the Management Information Format (MIF). This is a standard format developed by the DeskTop Management TaskForce (DMTF) industry consortium as part of the Desktop Management Interface (DMI). A short overview of DMTF and MIF is given in "The DMI Waiting Game" by Daniel Begun, in the Computer Shopper, n8, v15, p608, August 1995, and in "The End of the Balancing Act" by Barbara Gengler, in Internetwork, n8, v6, p47, August 1995. More information about MIF can be obtained from the DMTF at Hillsboro, Oregon, USA, or from their Web site, see in particular:

http://wwww.dmtf.org/general_info/factsheet.html, and
http://wwww.dmtf.org/techlinks/white_papers.html

Briefly, a MIF file, which is in the form of plain text, comprises multiple groups. Certain standard groups have been defined, some of which are mandatory, but user-defined groups are also permitted. Within each group is a list of named attributes and their values. For example, one required group is "ComponentID", which has the attributes of Manufacturer, Product, Version, Serial Number, Installation (date and time), and Verify (whether the corresponding component actually exists and is working). Referring back to Figure 2, a system supporting the DMI 280 has a MIF database 250, and a service layer 240 (these may be provided as part of the machine operating system). Manageable products (such as software applications) add or update their respective MIF files into or in the MIF database via a Component Interface (CI) of the service layer, whilst management applications can query the MIF database for information about the manageable applications via the Management Interface (MI) of the service layer. Samples of software MIF files plus further descriptive material can be obtained from the following Web site:

http://www-uk.hpl.hp.com/people/arp/dmtf/ver2.htm

The operation of the invention in one simple embodiment is shown schematically in Figure 3. Initially, the browser software detects the presence of an <APPLET> tag (step 310), representing a reference to an applet to be downloaded and executed. Next the browser looks for the MIFBASE and MIF attributes associated with the referenced applet (step 320). Assuming that these are located, the browser then accesses and retrieves the specified MIF file (steps 330 and 340). The MIF file is then displayed to the user in a window (step 350), allowing the user to scroll through the MIF information, and the user is prompted with a dialog box asking whether or not to continue with the applet (step 360). Assuming that the MIF information is acceptable, the user can then download and invoke the applet (steps 370 and 380). However, if the MIF file reveals some problem, such as the applet being extremely large, or requiring a more recent version of the Java virtual machine than is installed at the client, then the user has the option of effectively aborting, thereby avoiding needlessly downloading or trying to run the applet.

Returning to step 320, if the MIF and MIFBASE attributes are not located, then an attempt is made to download and invoke the applet directly. Note that this is also the path taken by a browser which does not recognise the MIF and MIFBASE attributes, which would be the case with a prior art browser.

Although the approach shown in Figure 3 is relatively simple, it has the drawback that the user has to scan through the MIF file itself. A more complex processing approach is depicted in the embodiment of Figure 4 (where we assume that the APPLET tag does contain the MIF/MIFBASE attributes). In this case, once the MIF information has been retrieved (step 405) from the server, it is loaded into the MIF database on the computer system (410) via the component interface (CI) of the service layer. Then, the management interface (MI) of the service layer is used to read out the MIF information from the MIF file (step 415). This interaction between the browser and the DMI is shown schematically in Figure 2.

The browser tests the MIF information against predetermined criteria (step 420), such as to determine whether the version number of the target operating system (the Java virtual machine) is compatible with the version installed at the client (note that the browser can use the DMI to retrieve only selected information from the MIF file, ignoring attributes that are not needed for this testing). Unless the predetermined criteria are satisfied, processing of the applet terminates, without the applet ever having been downloaded. There may be some criteria which are always or conditionally surfaced to the user for a decision; for example, if the applet is larger than a certain size, the user may be asked to confirm whether or not to continue (step 425), bearing in mind the length of time that may be required in order to download the applet. Assuming that the predetermined criteria are satisfied (with any necessary user confirmation), the applet is then downloaded (step 430). At this stage, further testing can be performed (step 435), typically for security purposes. For example, it can be verified that the applet is the correct size, and checks made that certain bytes specified in the MIF file properly match the content of the applet itself. Assuming that the predetermined criteria are once more

satisfied (again subject to any necessary user confirmation at step 440), then the applet can now be invoked (step 445). Note that if the browser subsequently deletes the applet from the client system, then generally it should update the MIF database accordingly.

It will be appreciated that a great many variations are possible on the processing illustrated in Figures 3 and 4. For example, the MIF file may simply be loaded into the MIF database (step 410) for audit purposes, without performing any checking against predetermined criteria, or asking for user verification. Typically this would be done after the applet has been downloaded (step 430), with the browser then attempting to invoke the applet (step 445), irrespective of the content of the MIF file. Alternatively, some browsers may do the testing against predetermined criteria all automatically, without any of the testing being surfaced to the user. Another possibility is that the MIF file could contain information needed to execute the applet. For example, it has been contemplated that applets could be downloaded in encrypted form, to reduce the possibility of tampering, in which case the MIF could be used to specify the algorithm needed to decrypt the applet.

It is optional whether the MIF file is loaded into the MIF database, and if so, whether the browser obtains the MIF information directly from the MIF file itself, or by using the MIF database. Loading the MIF file into the MIF database may be useful for audit purposes, and also allows the browser to exploit the MI for retrieving MIF information from the MIF file. This avoids the need for the browser to have to parse the MIF file to retrieve the values of the various attributes. However, if a large number of applets are downloaded onto the client machine, and have only a transitory existence there before being executed and deleted, then in some circumstances it may be preferable to obtain the MIF information directly from the MIF file itself, rather than having the overhead associated with interacting with the DMI.

Note that some of the MIF attributes discussed, such as size of the code, and version number of the target operating system, are already present in standard DMI groups, whilst others, such as type of encryption used, would need to be placed in user-defined groups. There are no firm limits on the type of information that can be contained within a MIF, although clearly it is desirable to have a set of predefined attributes, preferably as part of the DMI standard, which most or all browsers can interpret. It is also desirable that if a browser does not recognise particular attributes in this set, then it is still able to invoke the associated applet where possible (this is somewhat analogous to the situation where a browser does not recognise a particular html tag, but can still generally display the information on a page, even if not in exactly the desired format).

In the above embodiments, the MIF file is downloaded in its entirety for analysis at the client. However, an alternative approach is to retrieve the information concerning the applet by interrogating the associated MIF file at a remote location. In this case it is most likely that the MIF file is incorporated into a MIF database, so that the attributes MIF and MIFBASE would typically be used to locate the DMI through which access to the MIF database is provided. This arrangement is essentially the same as that shown in Figure 2, except that now the DMI 280 is located on a different machine from operating system 220 and browser 210.

There are several possible mechanisms for obtaining this remote access to the DMI. For example, it is intended that DMI itself should directly support such remote access via the MI, although a full implementation of this is not directly available yet. In this case, the processing might be similar to that shown in Figure 4, except that step 410 would be omitted, and the retrieval of the MIF information in step 415 would be performed via the remote access facility. Alternatively, the browser could use the well-known Simple Network Management Protocol (SNMP) to remotely interrogate the DMI, since DMI already supports this mechanism (the SNMP is described in "Internetworking with TCP/IP, Volume 2", by Douglas E Comer and David L Stevens, Prentice Hall, New Jersey, USA, 1991). Another possibility, which would avoid the need for the browser to include SNMP support, would be to have MIF and MIFBASE identify an intermediary program on the same machine as the MIF database. This intermediary would then receive requests in the http protocol from the browser for MIF information, with the requests specifying the particular desired attribute(s), submit these requests to the MI of the DMI, and then return the response to the browser, again in the http protocol.

The use of the MIF information directly at the remote location may be more efficient than initially copying the MIF information to the client, particularly if the browser is only interested in relatively few of the MIF attributes, since this avoids the need for the whole MIF file to be transmitted across the network. Of course, with this approach the browser would have to separately record any desired audit information.

It will be appreciated that although in the preceding description the identifier for the applet information has been broken down into two components designated MIFBASE and MIF, the identifier could be represented in any suitable form and under any appropriate name. Further, the information about the code need not be a MIF file, but could be in a different format, such as the SNMP management information base (MIB) format, or any other suitable format defined or specified by the user. Also, whilst the identifier in Example 2 is located within the APPLET tag itself, one could introduce a separate tag located between the <APPLET> and </APPLET> tags (ie analogous to the <PARAM> tag) in which to store the identifier. Of course, with all the above considerations, it is desirable to standardise on one particular option, which can then be supported by most or all browsers.

Further, whilst in the embodiments described above, the code for reviewing and interpreting the MIF file has been incorporated into the browser, this code could also form a separate program or object residing at the client, and known

to the browser. Thus on retrieving the MIF file, the browser would pass the MIF file to the MIF interpreter program, which would then analyze the file in the desired manner. For example, the interpreter program could simply open a window to display the MIF file, and possibly also ask the user whether to continue, corresponding to steps 350 and 360 in Figure 3; or simply add the MIF file to the local MIF database for auditing purposes, or test the MIF file against predetermined criteria. Where appropriate, the MIF interpreter program would return an appropriate response to the browser, indicating whether or not it should proceed with execution of the applet.

Alternatively, the MIF interpreter program could also be referenced by the Java virtual machine itself, which when invoked to run an applet, initially runs the MIF interpreter program, for example to record use of the applet for auditing purposes, or to ensure that the installed version of the virtual machine is compatible with that required for running the applet. Thus the MIF interpreter would act as a form of preprocessor in the virtual machine, in a similar manner (and immediately prior to) the byte code validator which currently initially verifies byte codes passed to the virtual machine for execution. In this approach the virtual machine preferably contains a hook to allow third party MIF interpreter programs to be bolted on to the virtual machine, although it would also be possible to incorporate that MIF interpreter function into the virtual machine itself.

Additionally, although the preferred embodiment has been described primarily with reference to an otherwise conventional computer workstation attached to the Internet, it will be appreciated that the invention is not so limited. For example, the computer workstation might be replaced by a so-called network computer, or any other hardware which is capable of receiving code over a network, such as a set-top box, a television Internet access device, and so on. Likewise, the relevant network may not be the Internet, but might be a corporate intranet (which typically uses Internet protocols such as html, but which is detached from the Internet itself), or any other suitable network. Further, whilst the code to be downloaded in the preferred embodiment comprises Java byte codes, this code might be in any appropriate language, and might be in a form either to be interpreted, for example by a virtual machine, as with Java, or to be directly executed on the hardware of the client machine. In this case the MIF file could be used to verify that the target operating system of the code to be downloaded matches that installed at the client machine.

Claims

1. A method of operating a computer attached to a network, having accessible at least one page of data that includes a reference to code that can be retrieved over the network for execution at said computer, characterised by the steps of:

   locating, associated with said reference, an identifier to information about said code; and

   retrieving the information about the code:

2. The method of claim 1, further comprising the step of utilising said information in order to determine whether or not to run said code.

3. The method of claim 2, further comprising the step of presenting at least part of said information to the user of said computer in order to allow said user to determine whether or not to run said code.

4. The method of claim 2 or 3, further comprising the step of comparing at least part of said information against one or more predetermined criteria to determine whether or not to run said code.

5. The method of any preceding claim, wherein said at least one page of data is in html format.

6. The method of claim 5, wherein said reference comprises an APPLET tag.

7. The method of claim 6, wherein said identifier comprises one or more attributes within said APPLET tag.

8. The method of claim 7, wherein said identifier comprises a first attribute specifying the server and directory where said information is located, and a second attribute specifying the name of a file containing the information.

9. The method of any preceding claim, wherein said computer runs a Web browser to access said at least one page of data, and said reference is in the form of a Universal Resource Locator (URL).

10. The method of any preceding claim, wherein the information is stored in a file, and said step of retrieving the

information about the code comprises copying the file onto said computer.

11. The method of claim 10, wherein the information is stored in a form conforming to the Management Information Format (MIF), and is loaded into the MIF database at the computer.

12. The method of claim 11, further comprising interrogating said MIF database to access the loaded MIF information.

13. The method of any of claims 1 to 9, wherein the information is stored in a remote database, and said step of retrieving the information comprises remotely interrogating said database over the network.

14. The method of any preceding claim, wherein said information includes the size of said code.

15. The method of any preceding claim, further comprising the steps of using said reference to retrieve the code over the network and executing the retrieved code at the computer.

16. A computer system, attachable to a network to access at least one page of data that includes a reference to code that can be retrieved over the network for execution at said computer, characterised by:

means for locating, associated with said reference, an identifier to information about said code; and

means for retrieving the information about the code.

17. A Web browser for accessing at least one page of data on the World Wide Web that includes a reference to applet code that can be retrieved over the Internet for execution at said computer, characterised by:

means for locating, associated with said reference, an identifier to information about said applet code; and

means for retrieving the information about the code.

18. The browser of claim 17, wherein said reference comprises an APPLET tag, and said identifier comprises one or more attributes within said APPLET tag.

19. A method of operating a server attached to a network, said method including steps of:

storing at least one page of data that includes a reference to code that can be retrieved over the network, and an identifier associated with said reference to indicate the location of information stored about said code;

receiving a request over the network for said at least one page of data;

and responsive to said request, transmitting said at least one page of data.

20. The method of claim 19, wherein said reference comprises an APPLET tag, and said identifier comprises one or more attributes within said APPLET tag.

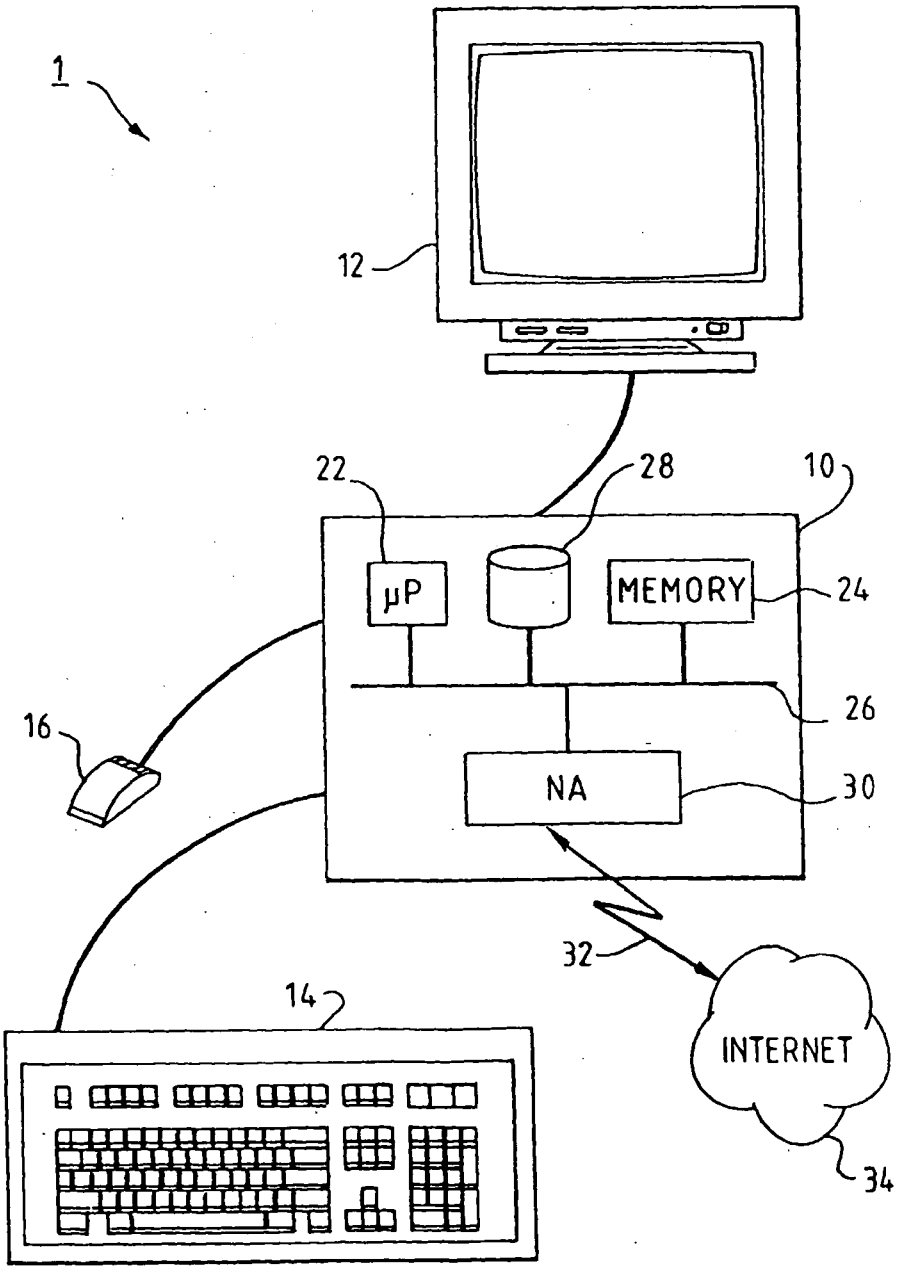


FIG. 1

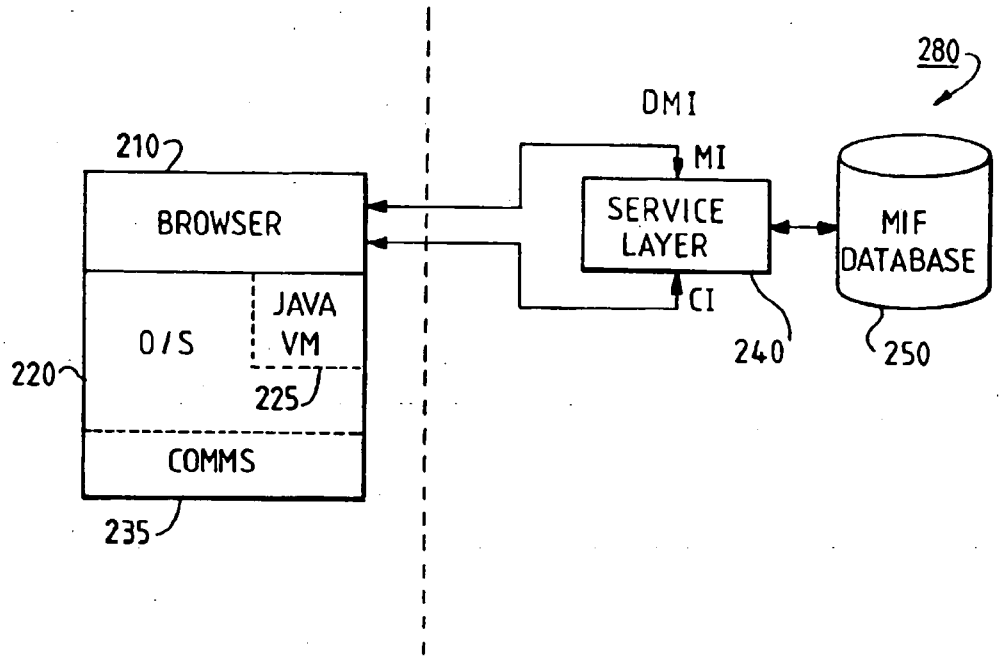

280
DMI
MI
210
BROWSER
SERVICE LAYER
MIF DATABASE
JAVA VM
O/S
220
225
CI
240
250
COMMS
235

FIG. 2

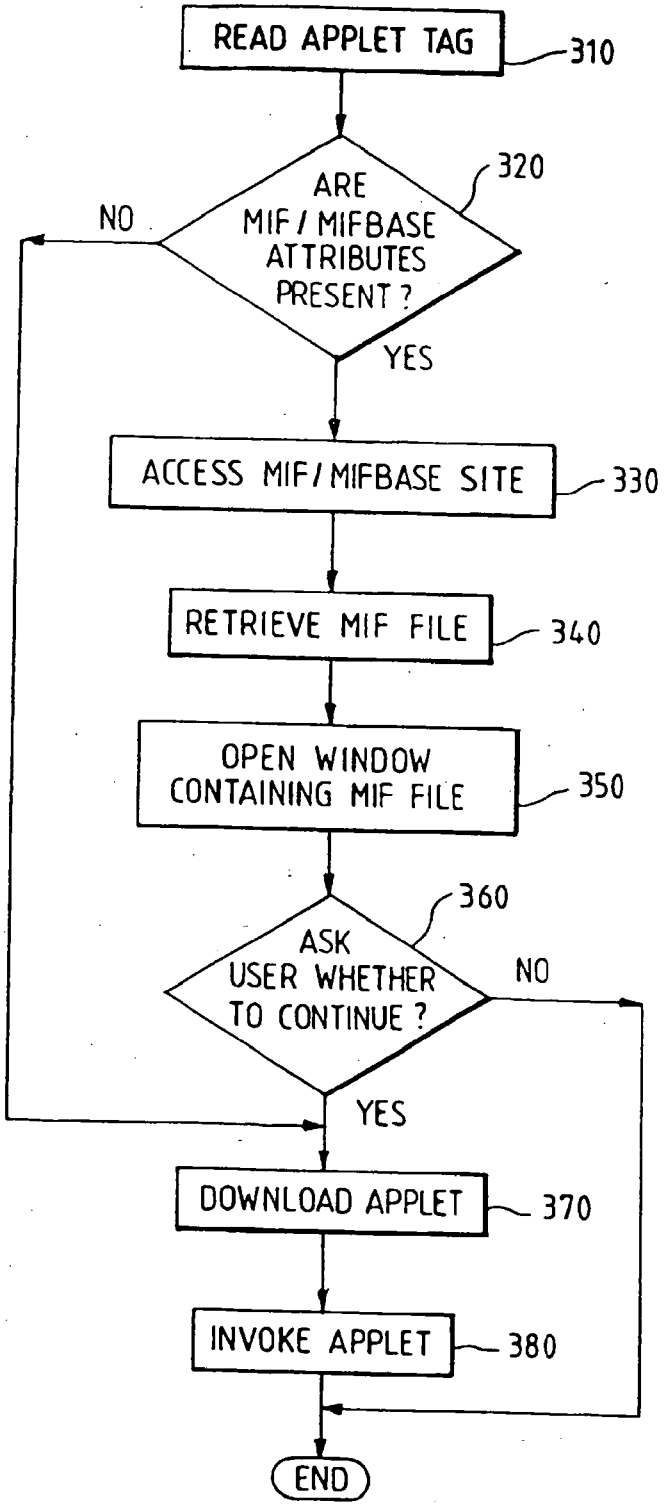

READ APPLET TAG
310
ARE MIF / MIFBASE ATTRIBUTES PRESENT ?
320
NO
YES
ACCESS MIF / MIFBASE SITE
330
RETRIEVE MIF FILE
340
OPEN WINDOW CONTAINING MIF FILE
350
ASK USER WHETHER TO CONTINUE ?
360
NO
YES
DOWNLOAD APPLET
370
INVOKE APPLET
380
END

FIG. 3

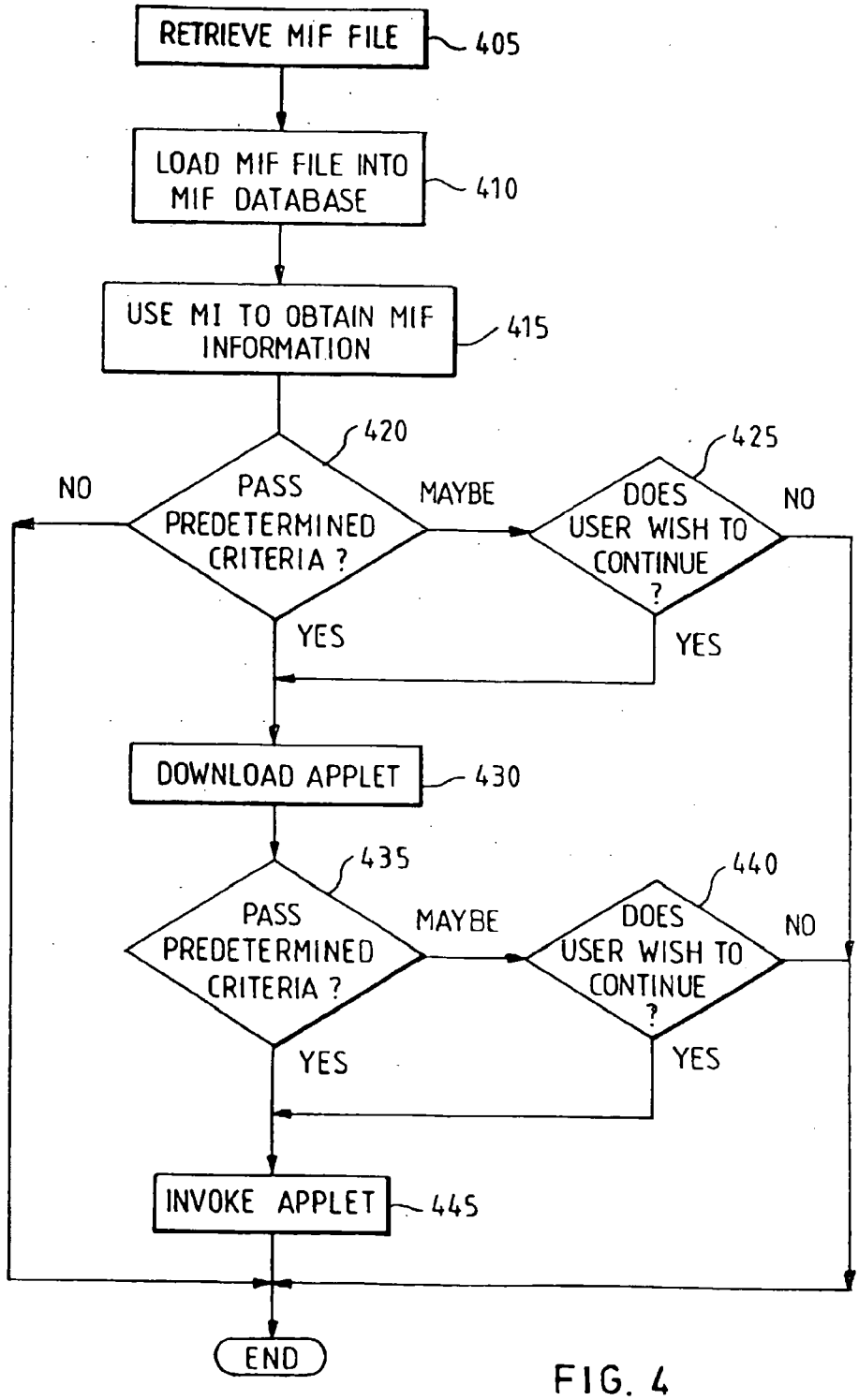

RETRIEVE MIF FILE
405
LOAD MIF FILE INTO MIF DATABASE
410
USE MI TO OBTAIN MIF INFORMATION
415
420
PASS PREDETERMINED CRITERIA ?
NO
MAYBE
425
DOES USER WISH TO CONTINUE ?
NO
YES
YES
DOWNLOAD APPLET
430
435
PASS PREDETERMINED CRITERIA ?
MAYBE
440
DOES USER WISH TO CONTINUE ?
NO
YES
YES
INVOKE APPLET
445
END

FIG. 4

(19)日本国特許庁（ＪＰ）　(12)公開特許公報(A)　(11)特許出願公開番号

特開平8－328583

(43)公開日　平成8年(1996)12月13日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ			技術表示箇所
Ｇ１０Ｌ	3/00	５６１		Ｇ１０Ｌ	3/00	５６１Ａ	
	5/06				5/06	Ｄ	

審査請求　有　請求項の数3　ＯＬ　（全４頁）

(21)出願番号　特願平7－136725

(22)出願日　平成７年(1995)６月２日

(71)出願人　000004237
日本電気株式会社
東京都港区芝五丁目７番１号

(72)発明者　坂井　信輔
東京都港区芝五丁目７番１号　日本電気株式会社内

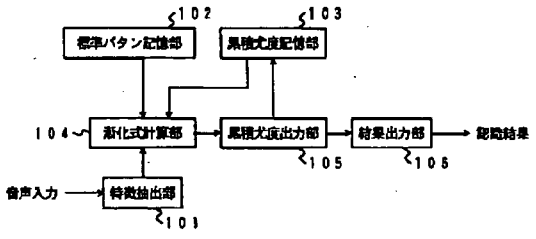
(74)代理人　弁理士　後藤　洋介　(外２名)

(54)【発明の名称】　音声認識装置

(57)【要約】

【目的】　周囲の雑音環境の変動やビーム幅の設定値の変化に対して安定した探索効率を有しながら、刈り取り閾値決定のための処理量が小さい、高速な音声認識装置を提供する。

【構成】　特徴抽出部１０１は、音声入力を特徴ベクトルの時系列に変換し、漸化式計算部１０４に出力する。標準パタン記憶部１０２は、標準パタンを記憶する。累積尤度記憶部１０３は、累積尤度出力部１０５から出力される累積尤度を記憶する。漸化式計算部１０４は、第ｉフレームの特徴ベクトル、標準パタン、及び第ｉ－１フレームまでの累積尤度から第ｉフレームまでの累積尤度を求める。累積尤度出力部１０５は、入力された累積尤度の集合から、あるＫフレームの部分区間では、第Ｍ位までの候補を選択、それ以降は、Ｋフレームの部分区間で求めた最尤候補と第Ｍ位の候補との累積尤度差の平均値を用いて候補を選択し、累積尤度記憶部１０３に出力する。結果出力部１０６は、最終フレームまでの累積尤度に基づいて認識結果を出力する。



1

【特許請求の範囲】

【請求項1】　入力音声の最初の数フレームで、最大尤度と一定個数番目の候補の尤度との差をもとめておき、それ以後は、前記差を用いて、ビームサーチにおける候補刈り取りのための閾値を設定することを特徴とする音声認識装置。

【請求項2】　音声信号を分析して特徴ベクトル時系列を出力する特徴抽出部と、
あらかじめ作成された標準パタンを蓄えておく標準パタン記憶部と、
累積尤度を保持する累積尤度記憶部と、
前記累積尤度記憶部に蓄えられた累積尤度と前記特徴ベクトルの時系列と前記標準パタンとから新しい累積尤度を求める漸化式計算部と、
前記特徴ベクトル時系列のうち、ある部分系列に対しては、漸化式計算部でもとめられた累積尤度のうち一定個数を出力するとともに、最大の累積尤度と前記一定個数番目の尤度との差を蓄積しておき、それ以降の部分系列に対しては、前記蓄積された尤度の差を用いて求められた閾値により、出力する累積尤度を決定する累積尤度出力部と、
前記累積尤度出力部から出力される累積尤度より前記音声信号に対する認識結果を求める結果出力部とを有することを特徴とする音声認識装置。

【請求項3】　入力音声の任意の個数の部分系列のおのおのに対して、第M位の候補の累積尤度の最大累積尤度との差の平均値を求め、次の部分系列の間では、前の部分系列で求めた前記差の平均値を用いて候補刈り取りの閾値を設定することを特徴とする音声認識装置における閾値設定方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、音声認識装置に関する。

【0002】

【従来の技術】音声認識装置は非常に大きい演算量を必要とするため、従来よりビームサーチによる演算量の削減が試みられている。ビームサーチによる候補刈り取りのためのビームの幅の設定法としては、各候補刈り取り時に、尤度の高いものから一定数の候補を残す方法と、最大尤度から一定幅の範囲の尤度をもつ候補を残す方法が良く知られている。

【0003】伊藤らによる、音響学会研究発表会講演論文集1993年10月73～74ページに掲載の論文「連続音声認識におけるビームサーチ」おいては、一定数の候補を残す方法のほうが、ビーム幅の設定値の変化に対して探索効率が安定していると報告されている。また、最大尤度から一定幅の範囲の尤度をもつ候補の数は、発声時の周囲雑音環境の影響を受けて変動すると考えられるが、一定数の候補を残す方法においては、その

2

ような候補数の変動はない。

【0004】

【発明が解決しようとする課題】しかしながら、上述の一定数（仮にM個とする）の候補を残す方法では、候補刈り取り時に第M位の候補を求めるための並べ替え処理が必要となるために、処理量が多いという欠点があった。

【0005】

【課題を解決するための手段】請求項1記載の発明によれば、入力音声の最初の数フレームで、最大尤度と一定個数番目の候補の尤度との差をもとめておき、それ以後は、前記差を用いて、ビームサーチにおける候補刈り取りのための閾値を設定することを特徴とする音声認識装置が得られる。

【0006】請求項2記載の発明によれば、音声信号を分析して特徴ベクトル時系列を出力する特徴抽出部と、あらかじめ作成された標準パタンを蓄えておく標準パタン記憶部と、累積尤度を保持する累積尤度記憶部と、前記累積尤度記憶部に蓄えられた累積尤度と前記特徴ベクトルの時系列と前記標準パタンとから新しい累積尤度を求める漸化式計算部と、前記特徴ベクトル時系列のうち、ある部分系列に対しては、漸化式計算部でもとめられた累積尤度のうち一定個数を出力するとともに、最大の累積尤度と前記一定個数番目の尤度との差を蓄積しておき、それ以降の部分系列に対しては、前記蓄積された尤度の差を用いて求められた閾値により、出力する累積尤度を決定する累積尤度出力部と、前記累積尤度出力部から出力される累積尤度より前記音声信号に対する認識結果を求める結果出力部とを有することを特徴とする音声認識装置が得られる。

【0007】請求項3記載の発明によれば、入力音声の任意の個数の部分系列のおのおのに対して、第M位の候補の累積尤度の最大累積尤度との差の平均値を求め、次の部分系列の間では、前の部分系列で求めた前記差の平均値を用いて候補刈り取りの閾値を設定することを特徴とする音声認識装置における閾値設定方法が得られる。

【0008】

【実施例】次に、本発明について図面を参照して説明する。

【0009】図1は、本発明の一実施例を示すブロック図である。図1を参照すると本発明の実施例は、特徴抽出部101と、標準パタン記憶部102と、累積尤度記憶部103と、漸化式計算部104と、累積尤度出力部105と、結果出力部106とから構成される。

【0010】特徴抽出部101は、音声入力を特徴ベクトルの時系列に変換し、漸化式計算部104に出力する。標準パタン記憶部102は、標準パタンを記憶する。累積尤度記憶部103は、累積尤度出力部105から出力される累積尤度を記憶する。処理が開始される以前には、全認識パス候補に対して累積尤度の初期値1.

3

0を保持する。漸化式計算部１０４は、第ｉフレームの特徴ベクトル、標準パタン、および第ｉ－１フレームまでの累積尤度から、第ｉフレームまでの累積尤度を求める。累積尤度出力部１０５は、入力された累積尤度の集合から、次フレームの累積尤度計算に用いられるものを選択し、累積尤度記憶部１０３に出力する。結果出力部１０６は、最終フレームまでの累積尤度に基づいて認識結果を出力する。

【００１１】次に、図１及び図２を参照して、本実施例の動作について説明する。

【００１２】入力された音声は、特徴抽出部１０１において、一定の時間間隔ごとに、音声の周波数をスペクトルをあらわす特徴ベクトルに変換され、漸化式計算部１０４に出力される。この一定の時間間隔を以下ではフレームと呼ぶ。第ｉフレームにおいて、漸化式計算部１０４では、標準パタン記憶部１０２に保持されている標準パタン

4

＊ＲＥＦ＝｛R_1，…，R_N｝、ここでR_v＝｛r_v（1），…，r_v（J_v）｝

を用いて、現在のフレームの特徴ベクトルの各標準パタンに対する局所的尤度

l_v（ｉ，ｊ）（ｗ＝1，…，Ｎ、ｊ＝1，…，J_v）

を求める。ここで、Ｎは標準パタン数、J_vはｗ番目の標準パタンのフレーム長である。次に、この局所的尤度、及び累積尤度記憶部１０３に保持されている第ｉ－１フレームの累積尤度集合

Ｇ＝｛g_1（ｉ－1，1），…，g_1（ｉ－1，J_1），…，g_N（ｉ－1，1），…，g_N（ｉ－1，J_N）｝

から、動的計画法に基づいた最大化処理により、下記数１として現在のフレームの認識パス候補およびその累積尤度を求める（図２のステップ1）。

【００１３】

【数１】

$$g_v(i,j)=\max_{x=0,1}(g_v(i-1,j-x)+l(i,j))$$

累積尤度出力部１０５は、あらかじめ決められたＫと比較して、ｉ≦Ｋであるならば、最大値から第Ｍ番目の累積尤度を求め、これを候補刈り取りのための閾値θとし、これと最大尤度との差ｄを求める。後で平均を求めるために、ｄの累積値S_dを、S_d＝S_d＋ｄと更新する（ステップ2，6、及び7）。

【００１４】なお、S_dは、第１フレーム以前には0に初期化しておく。

【００１５】ｉ＝Ｋの場合は、Ｋフレーム間の最大尤度と候補刈り取り閾値との差の平均Ｄ＝S_d／Ｋを求める（ステップ3）。

【００１６】また、ｉ＞Ｋの場合は、候補刈り取りのための閾値θは、θ＝g_{max}－Ｄとする。g_{max}は、第ｉフレームにおける累積尤度の最大値である（ステップ4）。

【００１７】各フレームにおいて、累積尤度出力部１０５は、累積尤度の閾値θよりも大きい尤度をもつ認識パス候補のみを累積尤度記憶部に出力する（ステップ5）。

【００１８】現フレームが最終フレームである場合は、累積尤度出力部１０５は、標準パタンの終端点に達したすべての認識パス候補を結果出力部１０６に出力する。結果出力部１０６は、累積候補が最大の認識パス候補をもとめ、認識結果を出力する（ステップ8，9）。

【００１９】以上、本実施例では、入力音声の最初のＫフレームで、第Ｍ位の候補の累積尤度の最大累積尤度との差の平均値を求めるという例によって説明したが、さらに一般には、入力音声の任意のL_{max}個の部分系列l_{i_1}，l_{i_2}，…，$l_{i_{L_{max}}}$（L_{max}≧1）（これらの部分系列を仮に学習区間と呼ぶ）のおのおのに対して、上記の平均値を求め、学習区間l_{i_k}と次の学習区間$l_{i_{k+1}}$の間では、l_{i_k}で求めた差の平均値を用いて候補刈り取りの閾値を設定するという方法をとることができる。

【００２０】

【発明の効果】以上説明したように、本発明による音声認識装置は、周囲の雑音環境の変動やビーム幅Ｍの設定値の変化に対応して第Ｍ位の候補の累積尤度と最大の累積尤度の差が大きく変動するような場合でも、入力の一部を用いてこの差の平均値を求めておき、これを用いて刈り取り閾値の決定を行なうので、入力の全ての区間に対して累積尤度第Ｍ位までの候補を残す方法に準ずる候補の刈り取りが行なわれ、安定した探索効率を有しながらも、刈り取り閾値決定のための処理量が多くならないという効果を有する。

【図面の簡単な説明】

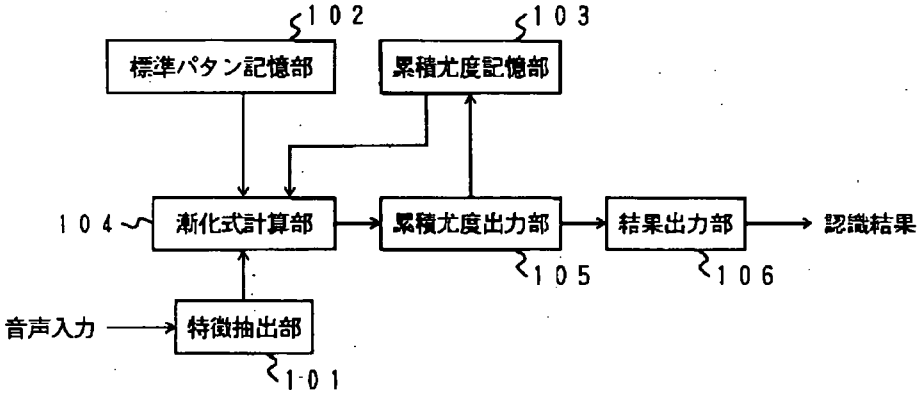
【図１】本発明の音声認識装置の一実施例の構成を示したブロック図である。

【図２】図１に示す音声認識装置の一実施例の処理の流れを示したフローチャートである。

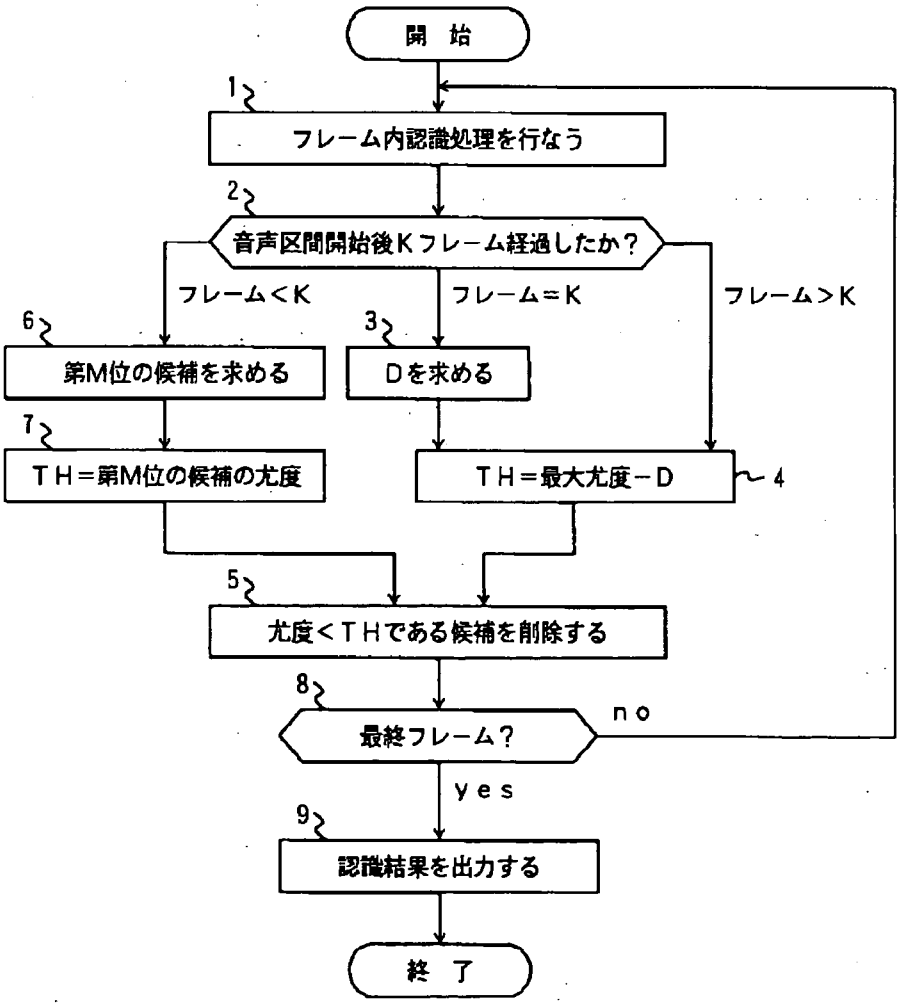
【符号の説明】

１０１　特徴抽出部
１０２　標準パタン記憶部
１０３　累積尤度記憶部
１０４　漸化式計算部
１０５　累積尤度出力部
１０６　結果出力部

【図１】


１０２
１０３
標準パタン記憶部
累積尤度記憶部
１０４
漸化式計算部
累積尤度出力部
結果出力部
認識結果
１０５
１０６
音声入力
特徴抽出部
１０１

【図２】


開　始
1
フレーム内認識処理を行なう
2
音声区間開始後Ｋフレーム経過したか？
フレーム＜Ｋ
フレーム＝Ｋ
フレーム＞Ｋ
6
第Ｍ位の候補を求める
3
Ｄを求める
7
ＴＨ＝第Ｍ位の候補の尤度
ＴＨ＝最大尤度－Ｄ
4
5
尤度＜ＴＨである候補を削除する
8
最終フレーム？
ｎｏ
ｙｅｓ
9
認識結果を出力する
終　了

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 0 865 180 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
16.09.1998 Bulletin 1998/38

(51) Int Cl.[6]: **H04L 12/56**, H04L 29/06, H04L 29/12

(21) Application number: **98301577.7**

(22) Date of filing: **03.03.1998**

(84) Designated Contracting States:
AT BE CH DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **14.03.1997 US 818989**

(71) Applicant: **LUCENT TECHNOLOGIES INC.**
Murray Hill, New Jersey 07974-0636 (US)

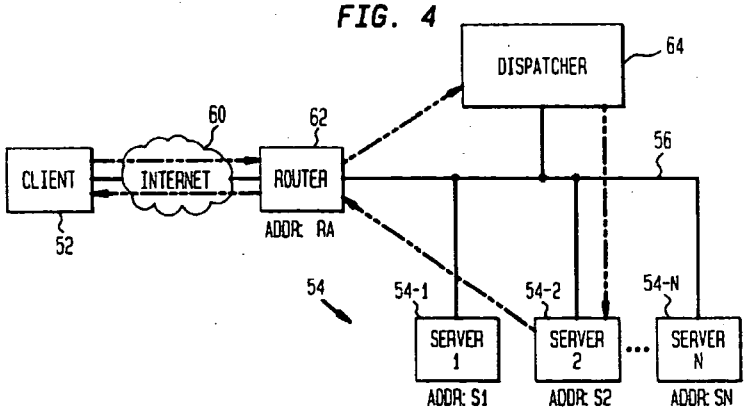
(72) Inventors:
- **Chung, Pi-Yu**
  Berkeley Heights, New Jersey 07922 (US)
- **Damani, Om P.**
  Austin, Texas 78705 (US)
- **Huang, Yennun**
  Bridgewater, New Jersey 08807 (US)
- **Kintala, Chandra M.**
  Warren, New Jersey 07059 (US)
- **Wang, Yi-Min**
  Berkeley Heights, New Jersey 07922 (US)

(74) Representative:
Watts, Christopher Malcolm Kelway, Dr.
Lucent Technologies (UK) Ltd,
5 Mornington Road
Woodford Green Essex, IG8 0TU (GB)

(54) Load distribution among servers in a TCP/IP network

(57) Methods and apparatus for hosting a network service on a cluster of servers, each including a primary and a secondary Internet Protocol (IP) address. A common cluster address is assigned as the secondary address to each of the servers in the cluster. The cluster address may be assigned in UNIX-based servers using an *ifconfig alias* option, and may be a ghost IP address that is not used as a primary address by any server in the cluster. Client requests directed to the cluster address are dispatched such that only one of the servers of the cluster responds to a given client request. The dispatching may use a routing-based technique, in which all client requests directed to the cluster address are routed to a dispatcher connected to the local network of the server cluster. The dispatcher then applies a hash function to the client IP address in order to select one of the servers to process the request. The dispatching may alternatively use a broadcast-based technique, in which a router broadcasts client requests having the cluster address to all of the servers of the cluster over a local network. The servers then each provide a filtering routine, which may involve comparing a server identifier with a hash value generated from a client address, in order to ensure that only one server responds to each request broadcast by the router.

FIG. 4



EP 0 865 180 A2

Printed by Jouve, 75001 PARIS (FR)

Description

Field of the Invention

The present invention relates generally to data communication networks such as the Internet and more particularly to techniques for hosting network services on a cluster of servers used to deliver data over a network in response to client requests, where the cluster of servers can be collectively identified by a client using a single-address image.

Background of the Invention

With the explosive growth of the World Wide Web, many popular Internet web sites are heavily loaded with client requests. For example, it has been reported in S. L. Garfinkel, "The Wizard of Netscape," Webserver Magazine, July/August 1996, pp. 59-63, that home pages of Netscape Communications receive more than 80 million client requests or "hits" per day. A single server hosting a service is usually not sufficient to handle this type of aggressive growth. As a result, clients may experience slow response times and may be unable to access certain web sites. Upgrading the servers to more powerful machines may not always be cost-effective. Another common approach involves deploying a set of machines, also known as a cluster, and configuring the machines to work together to host a single service. Such a server cluster should preferably publicize only one server name for the entire cluster so that any configuration change inside the cluster does not affect client applications. The World Wide Web and other portions of the Internet utilize an application-level protocol, known as the Hypertext Transfer Protocol (HTTP), which is based on a client/server architecture. The HTTP protocol is described in greater detail in "Hypertext Transfer Protocol -- HTTP/1.0," Network Working Group, May 1996, <http://www.ics.uci.edu/pub/ietf/http>, which is incorporated by reference herein.

FIG. 1 illustrates an exemplary client/server architecture suitable for implementing HTTP-based network services on the Internet. A client 12 generates an HTTP request for a particular service, such as a request for information associated with a particular web site, and a Transmission Control Protocol/Internet Protocol (TCP/IP) connection is then established between the client 12 and a server 14 hosting the service. The client request is delivered to the server 14 in this example via a TCP/IP connection over a first network 16, a router 18 and a second network 20. The first network 16 may be a wide area communication network such as the Internet, while the second network 20 may be an Ethernet or other type of local area network (LAN) interconnecting server 14 with other servers in a server cluster. The router 18, also referred to as a gateway, performs a relaying function between the first and second networks which is transparent to the client 12.

The client request is generated by a web browser or other application-layer program operating in an application layer 22-1 of the client 12, and is responded to by a file transfer system or other program in an application layer 22-2 of the server 14. The requested network service may be designated by a Unifonn Resource Locator (URL) which includes a domain name identifying the server 14 or a corresponding server cluster hosting the service. The application-level program of the client 12 initiates the TCP/IP connection by requesting a local or remote Domain Name Service (DNS) to map the server domain name to an IP address. The TCP and IP packet routing functions in client 12 and server 14 are provided in respective TCP layers 24-1, 24-2 and IP layers 26-1, 26-2. The TCP and IP layers are generally associated with the transport and network layers, respectively, of the well-known Open Systems Interconnection (OSI) model. The TCP layers 24-1, 24-2 process TCP packets of the client request and server response. The TCP packets each include a TCP header identifying a port number of the TCP connection between the client 12 and server 14. The IP layers 26-1, 26-2 process IP packets formed from the TCP packets of the TCP layers. The IP packets each include an IP header identifying an IP address of the TCP/IP connection between the client 12 and server 14.

The IP address for a given network service may be determined, as noted above, by the client accessing a conventional DNS. The IP layer 26-1 of the client 12 uses the resulting IP address as a destination address in the IP packet headers of client request packets. The IP address together with the TCP port number provide the complete transport address for the HTTP server process. The client 12 and server 14 also include data link and physical layers 28-1 for performing framing and other operations to configure client request or reply packets for transmission over the networks 16 and 20. The router 18 includes data link and physical layers 28-3 for converting client request and server reply packets to IP format, and an IP layer 26-3 for performing packet routing based on IP addresses. The server 14 responds to a given client request by supplying the requested information over the established TCP/IP connection in a number of reply packets. The TCP/IP connection is then closed.

There are many known techniques for distributing HTTP client requests to a cluster of servers. FIGS. 2 and 3 illustrate server-side single-IP-address image approaches which present a single IP address to the clients. An example of this approach is the TCP router approach described in D.M. Dias, W. Kish, R. Mukherjee and R. Tewari, "A Scalable and Highly Available Web Server," Proceedings of COMPCON '96, pp. 85-92, 1996, which is incorporated by reference herein. FIG. 2 illustrates the TCP router approach in which a client 12 establishes a TCP/IP connection over Internet 30 with a server-side router 32 having an IP address RA. The router 32 is connected via a LAN 36 to a server cluster 34 including N servers 14-i, i = 1, 2, ... N, having respec-

tive IP addresses S1, S2, ... SN. Each server of the cluster 34 generally provides access to the same set of contents, and the contents may be replicated on a local disk of each server, shared on a network file system, or served by a distributed file system.

The single-address image is achieved by publicizing the address RA of the server-side router 32 to the clients via the DNS. The client 12 therefore uses RA as a destination IP address in its request. The request is directed to the router 32, which then dispatches the request to a selected server 14-k of server cluster 34 based on load characteristics, as indicated by the dashed line connecting client 12 to server 14-k via router 32. The router 32 performs this dispatching function by changing the destination IP address of each incoming IP packet of a given client request from the router address RA to the address Sk of selected server 14-k. The selected server 14-k responds to the client request by sending reply packets over the established TCP/IP connection, as indicated by the dashed line connecting server 14-k to client 12. In order to make the TCP/IP connection appear seamless to the client 12, the selected server 14-k changes the source IP address in its reply packets from its address Sk to the router address RA. The advantages of this approach are that it does not increase the number of TCP connections, and it is totally transparent to the clients. However, since the above-noted source IP address change is performed at the IP layer in a given server, the kernel code of every server in the cluster has to be modified to implement this mechanism A proposed hybrid of the DNS approach and the TCP router approach, in which a DNS server selects one of several clusters of servers using a round-robin technique, suffers from the same problem.

FIG. 3 illustrates a server-side single-address image approach known as network address translation, as described in greater detail in E. Anderson, D. Patterson and E. Brewer, "The Magicrouter, an Application of Fast Packet Interposing," Symposium on Operating Systems Design and Implementation, OSDI, 1996, <http://www.cs.berkeley.edu/~eanders/magicrouter/ osdi96-mr-submission.ps>, and Cisco Local Director, <http://www.cisco.com/warp/public/751/lodir/index.html>, which are incorporated by reference herein. As in the TCP router approach of FIG. 2, the client 12 uses the router address RA as a destination IP address in a client request, and the router 32 dispatches the request to a selected server 14-k by changing the destination IP address of each incoming request packet from the router address RA to the address Sk of selected server 14-k. However, in the network address translation approach, the source IP addresses in the reply packets from the selected server 14-k.are changed not by server 14-k as in FIG. 2, but are instead changed by the router 32. The reply packet flow indicated by a dashed line in FIG. 2 thus passes from server 14-k to client 12 via router 32.

Compared to the TCP router approach of FIG. 2, network address translation has the advantage of server transparency. That is, no specific changes to the kernel code of the servers are required to implement the technique. However, both the TCP router and network address translation approaches require that the destination address in a request packet header be changed to a server address so that the server can accept the request. These approaches also require that the source address in a reply packet header be changed to the router address so that the client can accept the reply. These changes introduce additional processing overhead and unduly complicate the packet delivery process. In addition, because of the address changes, the above-described single-address image approaches may not be suitable for use with protocols that utilize IP addresses within an application, such as that described in K. Egevang and P. Francis, "The IP Network Address Translator," Network Working Group, RFC 1631, <http://www.safety.net/ rfc1631.txt>, which is incorporated by reference herein. Furthermore, in both the TCP router and network address translation approaches, the router 32 needs to store an IP address mapping for every IP connection. Upon receiving an incoming packet associated with an existing TCP connection, the router has to search through all of the mappings to determine which server the packet should be forwarded to. The router itself may therefore become a bottleneck under heavy load conditions, necessitating the use of a more complex hardware design, as in the above-cited Cisco Local Director.

It is therefore apparent that a need exists for improved techniques for hosting a network service on a cluster of servers while presenting a single-address image to the clients, without the problems associated with the above-described conventional approaches.

Summary of the Invention

The present invention provides methods and apparatus for hosting a network service on a cluster of servers. All of the servers in a server cluster configured in accordance with the invention may be designated by a single cluster address which is assigned as a secondary address to each server. All client requests for a web site or other network service associated with the cluster address are sent to the server cluster, and a dispatching mechanism is used to ensure that each client request is processed by only one server in the cluster. The dispatching may be configured to operate without increasing the number of TCP/IP connections required for each client request. The invention evenly distributes the client request load among the various servers of the cluster, masks the failure of any server or servers of the cluster by distributing client requests to the remaining servers without bringing down the service, and permits additional servers to be added to the cluster without bringing down the service. Although well-suited for use in hosting web site services, the techniques of the present invention may also be used to support a wide variety of other

server applications.

In an exemplary embodiment of the invention, a network service is hosted by a server cluster in which each server includes a primary IP address and a secondary IP address. A common cluster address is assigned as the secondary IP address for each of the servers. The cluster address may be an IP address which does not correspond to a primary IP address of any of the servers. In UNIX-based servers, the cluster address may be assigned as the secondary address for a given server using an *ifconfig alias* option. If a given server includes multiple network interface cards, the cluster address may be assigned to one of the network interface cards using a UNIX *ifconfig* command without the *alias* option, or other similar technique. A router is coupled to a local network of the server cluster and is also coupled via the Internet to a client. The router receives client requests from the Internet, and uses a dispatching technique to direct client requests having the cluster address as a destination. The client requests are dispatched such that each of the requests is processed by only one of the servers in the cluster. The dispatching function may be based on the result of applying a hash function to an IP address of the given client. A suitable hash function may be determined using an analysis of a distribution of client IP addresses in an access log associated with one or more of the servers. In the event that a server has failed, the hash function may be reapplied to the client IP address to identify another server.

Two illustrative dispatching techniques for providing a single-address image for a server cluster in accordance with the invention include routing-based dispatching and broadcast-based dispatching. In the routing-based technique, a dispatcher is coupled to the router and to a local network of the server cluster. The router directs client requests having the cluster address to the dispatcher, and the dispatcher selects a particular one of the servers to process a given client request based on the result of applying a hash function to the client address. In the broadcast-based technique, the router broadcasts client requests having the cluster address to each of the servers over the local network of the server cluster. Each of the servers implements a filtering routine to ensure that each client request is processed by only one of the servers. The filtering routine may involve applying a hash function to the client IP address associated with a given client request, and comparing the result to a server identifier to determine whether that server should process the client request.

The techniques of the present invention provide fast dispatching and can be implemented with reduced cost and complexity. The techniques are suitable for use in TCP/IP networks as well as networks based on a variety of other standards and protocols. Unlike the conventional single-address image approaches, the present invention does not require that a destination address in a request packet header be changed to a server address so that the server can accept the request, or that a source address in a reply packet header be changed to the router address so that the client can accept the reply. In addition, the router need not store an IP address mapping for every IP connection, nor is it required to search through such a mapping to determine which server a packet should be forwarded to. The router itself will therefore not become a bottleneck under heavy load conditions, and special router hardware designs are not required. These and other features and advantages of the present invention will become more apparent from the accompanying drawings and the following detailed description.

Brief Description of the Drawings

FIG. 1 is a block diagram illustrating a conventional client-server interconnection in accordance with the TCP/IP standard;

FIG. 2 illustrates a prior art TCP router technique for hosting a network service on a cluster of servers;

FIG. 3 illustrates a prior art network address translation technique for hosting a network service on a cluster of servers;

FIG. 4 illustrates a technique for hosting a network service on a cluster of servers using routing-based dispatching in accordance with an exemplary embodiment of the invention; and

FIG. 5 illustrates a technique for hosting a network service on a cluster of servers using broadcast-based dispatching in accordance with another exemplary embodiment of the invention.

Detailed Description of the Invention

The present invention will be illustrated below in conjunction with exemplary client/ server connections established over the Internet to a server cluster using the Transmission Control Protocol/Internet Protocol (TCP/IP) standard. It should be understood, however, that the invention is not limited to use with any particular type of network or network communication protocol. The disclosed techniques are suitable for use with a wide variety of other networks and protocols. The term "server cluster" as used herein refers to a group or set of servers interconnected or otherwise configured to host a network service. The terms "cluster address" and "single-address image" refer generally to an address associated with a group of servers configured to support a network service or services. A "ghost IP address" is one type of cluster address in the form of an IP address which is not used as a primary address for any server of a given server cluster. The term "network service" is intended to include web sites, Internet sites and data delivery services, as well as any other data transfer mechanism accessible by a client over a network. The term "client request" refers to a communication from a client which initiates the network service. A given client request may include multiple packets or only a single

packet, depending on the nature of the request.

The present invention provides an improved single-address image approach to distributing client requests to servers of a server cluster. In a preferred embodiment, the invention allows all servers of a server cluster to share a single common IP address as a secondary address. The secondary address is also referred to herein as a cluster address, and may be established using an *ifconfig alias* option available on most UNIX-based systems, or similar techniques available on other systems. The cluster address may be publicized to clients using the above-noted Domain Name Service (DNS) which translates domain names associated with Uniform Resource Locators (URLs) to IP addresses. All client requests to be directed to a service hosted by the server cluster are sent to the single cluster address, and dispatched to a selected one of the servers using routing-based or broadcast-based dispatching techniques to be described in greater detail below. Once a server is selected, future request packets associated with the same client request may be directed to the same server. All other communications within the server cluster may utilize primary IP addresses of the servers.

The above-noted *ifconfig alias* option is typically used to allow a single server to serve more than one domain name. For example, the *ifconfig alias* option allows a single server to attach multiple IP addresses, and thus multiple domain names, to a single network interface, as described in "Two Servers, One Interface" <http://www.thesphere.com/~dlp/TwoServers/>, which is incorporated by reference herein. Client requests directed to any of the multiple domain names can then be serviced by the same server. The server determines which domain name a given request is associated with by examining the destination address in the request packet. The present invention utilizes the *ifconfig alias* option to allow two servers to share the same IP address. Normally, two servers cannot share the same IP address because such an arrangement would cause any packet destined for the shared address to be accepted and responded to by both servers, confusing the client and possibly leading to a connection reset. Therefore, before a server is permitted to attach a new IP address to its network interface, a check may be made to ensure that no other server on the same local area network (LAN) is using that IP address. If a duplicate address is found, both servers are informed and warnings are issued. The routing-based or broadcast-based dispatching of the present invention ensures that every packet is processed by only one server of the cluster, such that the above-noted warnings do not create a problem.

An alternative technique for assigning a secondary address to a given server of a server cluster in accordance with the invention involves configuring the given server to include multiple network interface cards such that a different address can be assigned to each of the network interface cards. For example, in a UNIX-based system, conventional *ifconfig* commands may be used, without the above-described *alias* option, to assign a primary IP address to one of the network interface cards and a secondary IP address to another of the network interface cards. The secondary IP address is also assigned as a secondary IP address to the remaining servers in the cluster, and used as a cluster address for directing client requests to the cluster.

The exemplary embodiments of the present invention to be described below utilize dispatching techniques in which servers are selected based on a hash value of the client IP address. The hash value may be generated by applying a hash function to the client IP address, or by applying another suitable function to generate a hash value from the client IP address. For example, given N servers and a packet from a client having a client address CA, a dispatching function in accordance with the invention may compute a hash value *k* as CA *mod (N-1)* and select server *k* to process the packet. This ensures that all request or reply packets of the same TCP/IP connection are directed to the same server in the server cluster. A suitable hash function may be determined by analyzing a distribution of client IP addresses in actual access logs associated with the servers such that client requests are approximately evenly distributed to all servers. When a server in the cluster fails, the subset of clients assigned to that server will not be able to connect to it. The present invention addresses this potential problem by dynamically modifying the dispatching function upon detection of a server failure. If the hash value of a given client IP address maps to the failed server, the client IP address is rehashed to map to a non-failed server, and the connections of the remaining clients are not affected by the failure.

FIG. 4 illustrates a routing-based dispatching technique in accordance with the present invention. Solid lines indicate network connections, while dashed lines show the path of an exemplary client request and the corresponding reply. A client 52 sends a client request to a server cluster 54 including N servers 54-i, i = 1, 2, ... N having IP addresses S1, S2, ... SN and interconnected by an Ethernet or other type of LAN 56. The client request is formulated in accordance with the above-described HTTP protocol, and may include a URL with a domain name associated with a web site or other network service hosted by the server cluster 54. The client accesses a DNS to determine an IP address for the domain name of the service, and then uses the IP address to establish a TCP/IP connection for communicating with one of the servers 54-i of the server cluster 54. In accordance with the invention, a "ghost" IP address is publicized to the DNS as a cluster address for the server cluster 54. The ghost IP address is selected such that none of the servers 54-i of cluster 54 has that IP address as its primary address. Therefore, any request packets directed to the ghost IP address are associated with client requests for the service of the single-address image cluster 54. The use of the ghost IP address thus distin-

guishes a network service hosted by the cluster from activities of the servers 54-i which utilize the primary server addresses, and prevents interference with these primary address activities.

The client 52 uses the ghost IP address as a cluster address for directing its request to the server cluster 54. The request is directed over Internet 60 to a router 62 having an IP address RA. The router 62 includes a routing table having an entry or record directing any incoming request packets having the ghost IP address to a dispatcher 64 connected to the LAN 56. The dispatcher 64 includes an operating system configured to run in a router mode, using a routing algorithm which performs the dispatching described herein. In alternative embodiments, the functions of the dispatcher 64 could be incorporated into the router 62 in order to provide additional efficiency improvements. Each of the servers 54-i of the cluster 54 utilizes the above-described *ifconfig alias* option to set the ghost IP address as their secondary address. As noted above, this technique for setting a secondary address for each of the servers 54-i generally does not require any alteration of the kernel code running on the servers. In alternative embodiments, one or more of the servers 54-i may be configured to include multiple network interface cards, as previously noted, such that a different address can be assigned to each of the network interface cards of a given server using a UNIX *ifconfig* command or other similar technique.

The router 62 routes any packets having the ghost IP address to the dispatcher 64 in accordance with the above-noted routing table record. The dispatcher 64 then applies a hash function to the client IP address in a given request packet to determine which of the servers 54-i the given packet should be routed to. In the example illustrated in FIG. 4, the dispatcher 64 applies a hash function to the IP address of client 52 and determines that the corresponding request packet should be routed to server 54-2 at IP address S2. The dispatcher 64 then routes the request packet to the server 54-2 over LAN 56, as indicated by the dashed line, using the primary address S2 of server 54-2 to distinguish it from the other servers of cluster 54. After the network interface of server 54-2 accepts the packet, all higher level processing may be based on the ghost IP address because that is the destination address in the packet IP header and possibly in the application-layer packet contents. After processing the request, the server 54-2 replies directly to the client 52 via router 62 over the established TCP/IP connection, using the ghost IP address, and without passing through the dispatcher 64.

It should be noted that when a request packet destined for the ghost IP address is received by the network interface of dispatcher 64 and placed back onto the same network interface for delivery to one of the servers 54-i over LAN 56, it may cause an Internet control message protocol (ICMP) host redirect message to be sent to the router 62. This ICMP message is designed to direct the router 62 to update its routing table such that any future packets having the ghost IP address can bypass the dispatcher 64 and go directly to the destination server, as described in greater detail in W.R. Stevens, TCP/IP Illustrated, Vol. 1, Ch. 6, pp. 69-83, which is incorporated by reference herein. However, this effect is undesirable in the routing technique of FIG. 4 because the dispatcher 64 performs the server selection process as previously described. It therefore may be necessary to suppress the ICMP host redirect message for the ghost IP address by, for example, removing or altering the corresponding operating system code in the dispatcher. In the above-mentioned alternative embodiments in which the dispatching function is implemented within the router 62, the ICMP redirect message is not generated and therefore need not be suppressed. Another potential problem may arise when a reply packet is sent back to the client 52 from the selected server 54-2 with the ghost IP address, in that it may cause the router 62 to associate, in its Address Resolution Protocol (ARP) cache, the ghost IP address with the LAN address of the selected server. The operation of the ARP cache is described in greater detail in W.R. Stevens, TCP/IP Illustrated, Vol. 1, Chs. 4 and 5, pp. 53-68, which is incorporated by reference herein. The illustrative embodiment of FIG. 4 avoids this problem by automatically routing the request packets to the dispatcher 64, and then dispatching based on the server primary IP address, such that the router ARP cache is not used.

FIG. 5 illustrates a broadcast-based dispatching technique in accordance with the present invention. Again, solid lines indicate network connections, while dashed lines show the path of an exemplary client request and the corresponding reply. As in the FIG. 4 routing-based embodiment, client 52 sends a client request to server cluster 54 including N servers 54-i, i = 1, 2, ... N connected to LAN 56 and having IP addresses S1, S2, ... SN. The client 52 uses the above-described ghost address as a cluster address for directing its request to the server cluster 54. The request is directed over Internet 60 to a router 70 having an IP address RA. The router 70 broadcasts any incoming request packets having the ghost IP address to the LAN 56 interconnecting the servers 54-i of the server cluster 54, such that the request packet is received by each of the servers 54-i.

Each of the servers 54-i of the cluster 54 implements a filtering routine in order to ensure that only one of the servers 54-i processes a given client request. The filtering routine may be added to a device driver of each of the servers 54-i. In an exemplary implementation, each of the servers 54-i is assigned a unique identification (ID) number. The filtering routine of a given server 54-i computes a hash value of the client IP address and compares it to the ID number of the given server. If the hash value and the ID number do not match, the filtering routine of the given server rejects the packet. If the hash value and the ID number do match, the given server accepts and processes the packet as if it had received the packet through a conventional IP routing mechanism. In

the illustrative example of FIG. 5, a packet associated with request from client 52 is broadcast by the router 70 to each of the servers 54-i of the server cluster 54 over the LAN 56 as previously noted. The filtering routine of server 54-2 generates a hash value of the client IP address which matches the unique ID number associated with server 14-2, and server 14-2 therefore accepts and processes the packet. The filtering routines of the N-1 other servers 54-i each indicate no match between the client IP address and the corresponding server ID number, and therefore discard the broadcast packet. The reply packets are sent back to the client 52 via router 70, as indicated by the dashed lines, using the ghost IP address.

The broadcast-based dispatching technique of FIG. 5 may be implemented using a permanent ARP entry within the router 70, to associate the ghost IP address with the Ethernet or other local network broadcast address associated with LAN 56 of the cluster 54. A potential problem is that any reply packet from a selected server appears to be coming from the ghost IP address, and may therefore cause the router 70 to overwrite the entry in its ARP cache such that the ghost IP address is associated with the LAN address of the selected server. This potential problem may be addressed by setting up a routing table entry in the router 70 to direct all packets having a ghost IP destination address to a second ghost IP address which is a legal subnet address in the LAN 56 of the server cluster 54 but is not used by any server. In addition, an entry is inserted in the ARP cache of the router 70 to associate the second ghost IP address with the broadcast address of the LAN 56. When the router 70 routes a packet to the second ghost IP address, it will then actually broadcast the packet to each of the servers 54-i of the cluster 54. Since no reply packet is sent from the second ghost IP address, the corresponding entry of the router ARP cache will remain unchanged. Another potential problem is that some operating systems, such as the NetBSD operating system, do not allow a TCP packet to be processed if it is received from a broadcast address. This potential problem may be avoided by a suitable modification to the broadcast address in the LAN packet header attached to the packet.

The routing-based and broadcast-based dispatching techniques described in conjunction with FIGS. 4 and 5 above have been implemented on a cluster of Sun SPARC workstations. The NetBSD operating system, as described in NetBSD Project, <http:/www. NetBSD. org>, was used to provide any needed kernel code modifications. The dispatching overhead associated with both techniques is minimal because the packet dispatching is based on simple IP address hashing, without the need for storing or searching any address-mapping information. In the routing-based dispatching technique, the additional routing step in the dispatcher 64 typically adds a delay of about 1 to 2 msecs to the TCP round-trip time of each incoming request packet. A study in W. R. Stevens, TCP/IP Illustrated, Volume 3, pp. 185-186, which is incorporated by reference herein, indicates that the median TCP round-trip time is 187 msecs. The additional delay attributable to the routing-based dispatching is therefore negligible. Although the additional routing step for every request packet sent to the ghost IP address may increase the traffic in the LAN of the server cluster, the size of a request in many important applications is typically much smaller than that of the corresponding response, which is delivered directly to the client without the additional routing. In the broadcast-based dispatching technique, the broadcasting of each incoming request packet on the LAN of the server cluster does not substantially increase network traffic. Although a hash value is computed for each incoming packet having the ghost IP destination address, which increases the CPU load of each server, this additional computation overhead is negligible relative to the corresponding communication delay.

Both the routing-based and broadcast-based dispatching techniques of the present invention are scalable to support relatively large numbers of servers. Although the dispatcher in the routing-based technique could present a potential bottleneck in certain applications, a study in the above-cited D.M. Dias et al. reference indicates that a single dispatcher can support up to 75 server nodes, which is sufficient support for many practical systems. The number of servers supported may be even higher with the present invention given that the routing-based dispatching functions described herein are generally simpler than those in the D.M. Dias et al. reference. It should also be noted that additional scalability can be obtained by combining the routing-based dispatching of the present invention with a DNS round-robin technique. For example, a DNS server may be used to map a domain name to one of a number of different ghost IP addresses belonging to different server clusters using a round-robin technique. In the broadcast-based dispatching technique, there is no potential dispatching bottleneck, although the device drivers or other portions of the servers may need to be modified to provide the above-described filtering routines.

The routing-based and broadcast-based dispatching of the present invention can also provide load balancing and failure handling capabilities. For example, given N servers and a packet from client address CA, the above-described routing-based dispatching function may compute a hash value k as $CA \bmod (N-1)$ and select server k to process the packet. More sophisticated dispatching functions can also be used, and may involve analyzing the actual service access log to provide more effective load balancing. In order to detect failures, each server may be monitored by a watchdog daemon such as the *watchd* daemon described in greater detail in Y. Huang and C. Kintala, "Software Implemented Fault Tolerance: Technologies and Experience," Proceedings of the 23rd International Symposium on Fault-Tolerant Computing - FTCS, Toulouse, France, pp. 2-9, June

1993, which is incorporated by reference herein. When a server fails, the corresponding *watchd* daemon initiates a change of the dispatching function to mask the failure and rebalance the load. A system call interface may be implemented to allow the dispatching function to be changed while the servers remain on-line. In routing-based dispatching, the *watchd* daemon may notify the dispatcher to change the dispatching function, while in broadcast-based dispatching, all servers may be notified to modify their filtering routines. For example, if a server *k* fails, the new dispatching function may check to see if the hash value *CA mod N* equals *k*. If it does, a new hash value $j = CA \bmod (N-1)$ is computed. If *j* is less than *k*, the packet goes to server *j*. Otherwise, the packet goes to server *j+1*. This technique does not affect the clients of non-failed servers, reassigns the clients of the failed server evenly to the remaining servers, and can be readily extended to handle multiple server failures. Additional servers can be added to the cluster without bringing down the service by changing the dispatching function from *CA mod N* to *CA mod (N+1)*.

In routing-based dispatching, the dispatcher may become a single point of failure, and therefore should also be monitored by a *watchd* daemon or other suitable failure monitoring mechanism. Upon detecting a failure, the *watchd* daemon may trigger a transfer of the dispatching function from the primary dispatcher to a backup dispatcher, and then direct the router to change the entry in its routing table such that future incoming request packets are routed to the backup dispatcher. Since no mapping table is maintained by the primary dispatcher, this approach is substantially stateless. Proper routing may be ensured by simply utilizing consistent routing functions in the primary and backup dispatchers, without the substantial additional costs associated with mapping-based approaches.

The use of the *ifconfig alias* option or other similar technique to provide a single-address image for a server cluster provides a number of advantages over the conventional techniques described previously. For example, it avoids the need to change the destination address in a request packet header so that a particular server can accept the request, and the need to change the source address in a reply packet header to the cluster address so that the client can accept the reply. With the single-address image approach of the present invention, all servers can accept and respond to packets having the cluster address, so that the addresses in the request and reply packet headers do not need to be modified. Since the single-image approach of the present invention does not require alternation of the packet addresses, it is suitable for use with a wide variety of protocols, including those protocols which utilize IP addresses within an application program. In addition, the single-address image approach of the present invention does not require a router to store or to search through a potentially large number of IP address mappings in order to determine which cluster server should receive a request packet. The invention thus effectively removes the possibility that the router may become a bottleneck under heavy load conditions.

The above-described embodiments of the invention are intended to be illustrative only. Numerous alternative embodiments may be devised by those skilled in the art without departing from the scope of the following claims.

Claims

1. A method of routing client requests to a plurality of servers configured to support a network service over a communication network, each of the servers having a primary address, the method comprising the steps of:

   assigning a common address as a secondary address for each of the plurality of servers; and processing client requests directed to the common address such that each of the requests is processed by a particular one of the plurality of servers.

2. The method of claim 1 wherein the network utilizes a TCP/IP protocol and the primary and secondary addresses are primary and secondary IP addresses, respectively.

3. The method of claim 2 wherein the common address is an IP address which does not correspond to a primary IP address of any of the plurality of servers.

4. The method of any of the preceding claims wherein at least one of the plurality of servers is a UNIX-based server including multiple network interface cards, and the assigning step includes assigning the common address for the at least one server using an *ifconfig* command..

5. The method of any of claims 1 to 3 wherein the plurality of servers are UNIX-based servers, and the assigning step includes assigning the common address utilizing an *ifconfig alias* option for at least a subset of the plurality of servers.

6. The method of any of the preceding claims wherein the processing step includes the step of dispatching a request of a given client to one of the plurality of servers based on application of a hash function to an IP address of the given client.

7. The method of claim 6 wherein the hash function is determined based on an analysis of a distribution of client IP addresses in an access log associated with one or more of the servers.

8. The method of claim 6 wherein the dispatching step includes reapplying the hash function to the client IP address to identify another servers if a server identified as a results of a previous application of the hash function has failed.

9. The method of any of claims 1 to 5 wherein the processing step includes the steps of:

routing client requests directed to the common address to a dispatcher connected to a local network associated with the plurality of servers; and
selecting a particular one of the servers to process a given client request based on application of a hash function to a corresponding client address in the dispatcher.

10. The method of any of claims 1 to 5 wherein the processing step includes the steps of:

broadcasting a given client request directed to the common address to each of the plurality of servers over a local network associated with the servers; and
implementing a filtering routine in each of the plurality of servers so that the given client request is processed by only one of the servers.

11. The method of claim 10 wherein the implementing step includes the steps of:

applying a hash function to a client IP address associated with the given client request; and
comparing the result of the applying step to an identifier of a particular server to determine whether that server should process the given client request.

12. An apparatus for routing client requests to a plurality of servers configured to support a network service over a communication network, each of the servers having a primary address, the apparatus comprising:

means for assigning a common address as a secondary address for each of the plurality of servers; and
means for processing client requests directed to the common address such that each of the requests is processed by a particular one of the plurality of servers.

13. The apparatus of claim 12 wherein the processing means is operative to dispatch a request of a given client to one of the plurality of servers based on application of a hash function to an IP address of the given client.

14. The apparatus of claim 13 wherein the hash function is determined based on an analysis of a distribution of client IP addresses in an access log associated with one or more of the servers.

15. The apparatus of claim 13 wherein the processing means is further operative to reapply the hash function to the client IP address to identify another server if a server identified as a result of a previous application of the hash function has failed.

16. The apparatus of any of claims 12 to 15 wherein the processing means further includes a dispatcher connected to a local network associated with the plurality of servers, wherein the dispatcher is operative to receive client requests directed to the common address, and to select a particular one of the servers to process a given client request based on application of a hash function to a corresponding client address.

17. The apparatus of any of claims 12 to 15 wherein the processing means further includes:

means for broadcasting a given client request directed to the common address to each of the plurality of servers over a local network associated with the servers; and
means for filtering the given client request in each of the plurality of servers so that the given client request is processed by only one of the servers.

18. The apparatus of claim 17 wherein the filtering means is operative to apply a hash function to a client IP address associated with the given client request, and to compare the result of the applying step to an identifier of a particular server to determine whether that server should process the given client request.

19. An apparatus for routing client requests for a network service over a communication network, the apparatus comprising:

a plurality of servers configured to support the network service, each of the servers having a primary address and a secondary address, wherein a common address is assigned as the secondary address for each of the plurality of servers; and
a router coupled to the servers and operative to route client requests directed to the common address such that each of the requests is processed by a particular one of the plurality of servers.

20. The apparatus of claim 19 wherein the router is fur-

ther operative to route client requests such that a request of a given client is routed to one of the plurality of servers based on application of a hash function to an IP address of the given client.

21. The apparatus of claim 20 wherein the hash function is determined based on an analysis of a distribution of client IP addresses in an access log associated with one or more of the servers.

22. The apparatus of claim 20 wherein the hash function is reapplied to the client IP address to identify another server if a server identified as a result of a previous application of the hash function has failed.

23. The apparatus of any of claims 19 to 22 further including a dispatcher coupled to the router and to a local network associated with the plurality of servers, such that the router directs client requests having the common address to the dispatcher, and the dispatcher selects a particular one of the servers to process a given client request based on application of a hash function to a corresponding client address.

24. The apparatus of any of claims 19 to 22 wherein the router is further operative to broadcast a given client request directed to the common address to each of the plurality of servers over a local network associated with the servers, and further wherein each of the servers implements a filtering routine so that the given client request is processed by only one of the servers.

25. The apparatus of claim 24 wherein the filtering routine involves applying a hash function to a client IP address associated with the given client request. and comparing the result to an identifier of a particular server to determine whether that server should process the given client request.

26. The apparatus of any of claims 12 to 25 wherein the network utilizes a TCP/IP protocol and the primary and secondary addresses are primary and secondary IP addresses, respectively.

27. The apparatus of claim 26 wherein the common address is an IP address which does not correspond to a primary IP address of any of the plurality of servers.

28. The apparatus of any of claims 12 to 27 wherein at least one of the plurality of servers is a UNIX-based server including multiple network interface cards, and the common address is assigned for the at least one server using an *ifconfig* command.

29. The apparatus of any of claims 12 to 27 wherein the plurality of servers are UNIX-based servers, and the common address is assigned as the secondary address of the plurality of servers by utilizing an *ifconfig alias* option for at least a subset of the plurality of servers.

FIG. 1
(PRIOR ART)

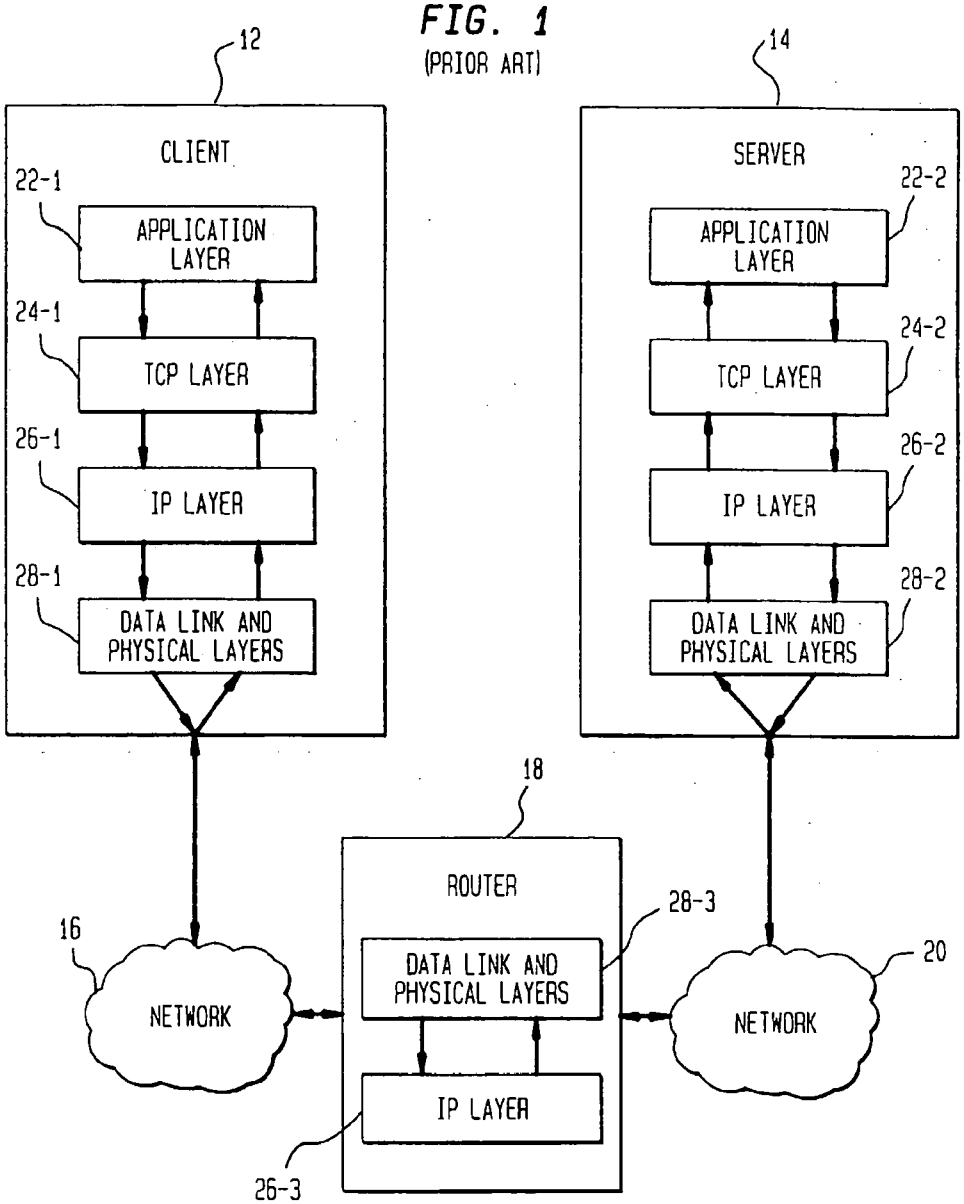


FIG. 2
(PRIOR ART)

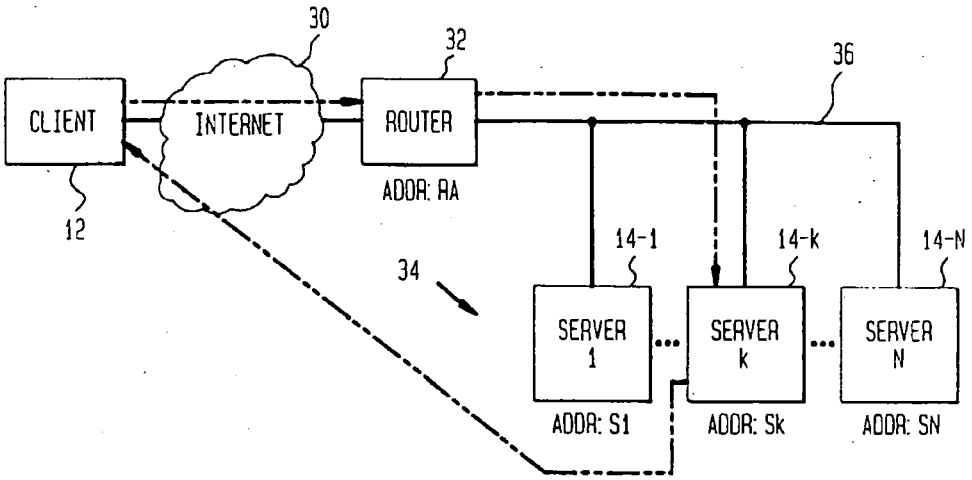


FIG. 3
(PRIOR ART)

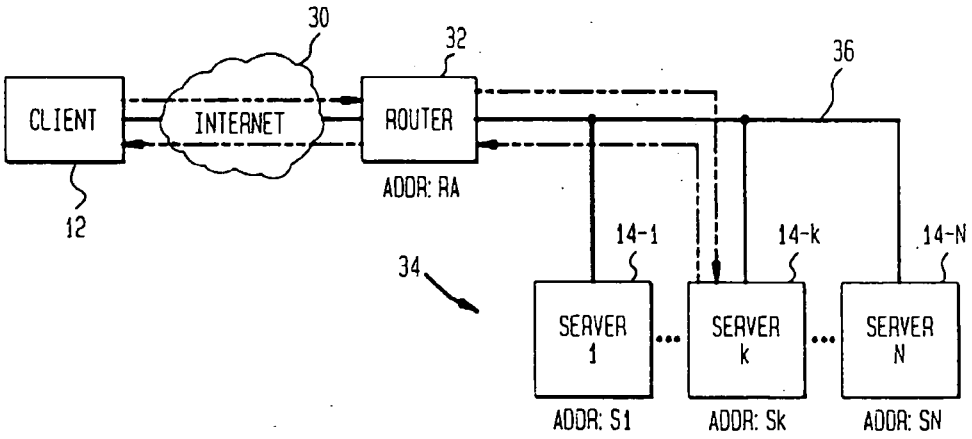


FIG. 4

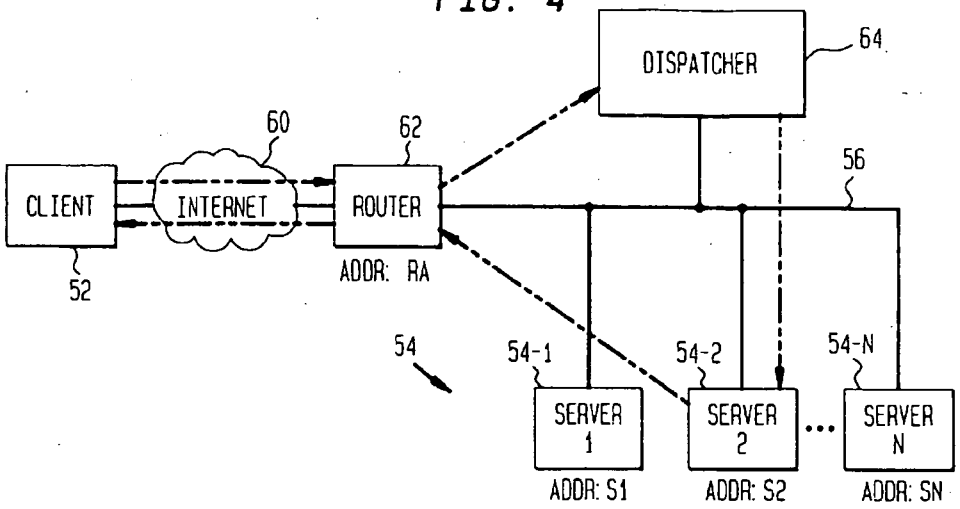
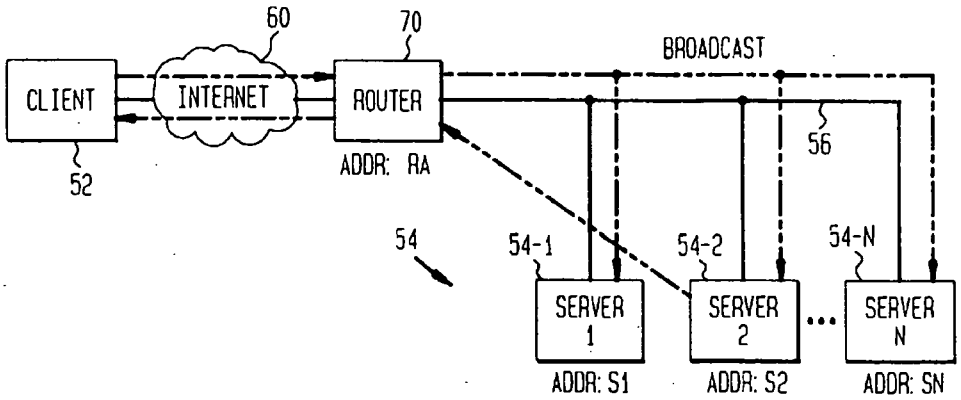

DISPATCHER
64
60
62
56
CLIENT
INTERNET
ROUTER
ADDR: RA
52
54
54-1
54-2
54-N
SERVER 1
SERVER 2
SERVER N
ADDR: S1
ADDR: S2
ADDR: SN

FIG. 5


60
70
BROADCAST
CLIENT
INTERNET
ROUTER
ADDR: RA
56
52
54
54-1
54-2
54-N
SERVER 1
SERVER 2
SERVER N
ADDR: S1
ADDR: S2
ADDR: SN

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号

特開平10－27148

(43)公開日　平成10年(1998)１月27日

(51)Int.Cl.[6]		識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ	13/00	３５７		Ｇ０６Ｆ	13/00	３５７Ｚ
	15/16	４７０			15/16	４７０Ｊ
Ｈ０４Ｌ	12/56		9744－5Ｋ	Ｈ０４Ｌ	11/20	１０２Ｄ

審査請求　未請求　請求項の数11　ＯＬ　（全 11 頁）

(21)出願番号　特願平8－180722

(22)出願日　平成８年(1996)７月10日

(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地

(72)発明者　大野　修司
神奈川県川崎市幸区鹿島田890番地　株式会社日立製作所情報・通信開発本部内

(72)発明者　後藤　法宏
神奈川県横浜市戸塚区戸塚町5030番地　株式会社日立製作所ソフトウェア開発本部内

(72)発明者　小林　一司
神奈川県海老名市下今泉810番地　株式会社日立製作所オフィスシステム事業部内

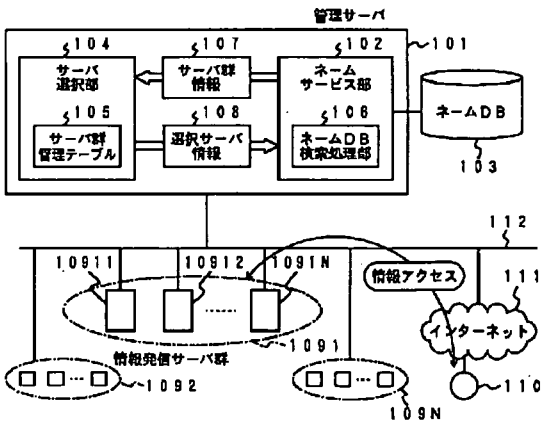
(74)代理人　弁理士　富田　和子

最終頁に続く

(54)【発明の名称】　インターネット用サーバシステム

(57)【要約】
【課題】端末利用ユーザに特別な操作を強いること無く、サーバ群内におけるアクセス分散と障害発生時等のバックアップが図れるインターネット用サーバシステムを提供することにある。
【解決手段】サーバ群１０９１に対して与えられたホスト名と、サーバ群１０９１を構成する複数台のサーバのそれぞれの通信用アドレスと、これらの通信用アドレスの中から目的の通信用アドレスを選択する際に用いられる選択情報とが格納されたサーバ群管理テーブル１０５と、インターネット１１１を介して、前記ホスト名に対応する通信用アドレスを要求されると、前記選択情報の内容に応じて、サーバ群管理テーブル１０５に格納されている通信用アドレスの何れかを選択するサーバ選択１０４と、サーバ選択部１０４が選択した通信用アドレスを通信用アドレスの要求元に送信するネームサービス部１０２とを備える。

図１



【特許請求の範囲】
【請求項1】インターネットを介して情報発信を行う複数台のサーバと、該複数台のサーバのそれぞれを管理する管理サーバとを備えたサーバシステムにおいて、
前記複数台のサーバの少なくとも一部は、同一情報の発信を行う複数台のサーバから成るサーバ群を構成し、
前記管理サーバは、
前記サーバ群に対して与えられたホスト名と、前記サーバ群を構成する前記複数台のサーバのそれぞれの通信用アドレスと、これらの通信用アドレスの中から目的の通信用アドレスを選択する際に用いられる選択情報とが格納された管理テーブルと、
前記インターネットを介して、前記ホスト名に対応する通信用アドレスを要求されると、前記選択情報の内容に応じて、前記管理テーブルに格納されている通信用アドレスの何れかを選択する選択手段と、
前記選択手段が選択した通信用アドレスを前記通信用アドレスの要求元に送信するネームサービス手段とを備えたことを特徴とするサーバシステム。
【請求項2】請求項1において、
前記選択情報には、前記通信用アドレスを選択するための選択順序が含まれており、
前記選択手段は、前記選択情報の前記選択順序にしたがって前記通信用アドレスを選択することを特徴とするサーバシステム。
【請求項3】請求項1において、
前記管理サーバは、各サーバの負荷状況を判断し、その判断結果を前記選択情報の一部として前記管理テーブルに格納する負荷監視手段をさらに備え、
前記選択手段は、前記通信用アドレスを選択する際に、前記判断結果を用いて、その時点で負荷が最も少ないサーバを選び、そのサーバの通信用アドレスを選択することを特徴とするサーバシステム。
【請求項4】請求項1において、
前記各サーバには、そのサーバの負荷状況を判断して、その結果を判断情報として前記管理サーバに送信する手段が設けられており、
前記管理サーバには、前記各サーバから送信されたそれぞれの判断情報を前記選択情報の一部として前記管理テーブルに格納する手段が設けられており、
前記選択手段は、前記通信用アドレスを選択する際に、前記判断情報を用いて、その時点で負荷が最も少ないサーバを選び、その通信用アドレスを選択することを特徴とするサーバシステム。
【請求項5】請求項1、2、3または4において、
前記サーバ群は、複数群構成されており、
前記管理テーブルには、前記各サーバ群に対して与えられたホスト名と、前記各サーバ群を構成する前記複数台のサーバのそれぞれの通信用アドレスと、これらの通信用アドレスの中から目的の通信用アドレスを選択する際に用いられる選択情報とが格納され、
前記選択手段は、前記インターネットを介して、前記ホスト名に対応する通信用アドレスを要求されると、前記選択情報の内容に応じて、該ホスト名で示されるサーバ群に含まれている各サーバの通信用アドレスの中から目的の通信用アドレスを選択することを備えたことを特徴とするサーバシステム。
【請求項6】インターネットを介して情報発信を行う複数台のサーバと、該複数台のサーバのそれぞれを管理する管理サーバとを備えたサーバシステムにおいて、
前記複数台のサーバの少なくとも一部は、同一情報の発信を行う複数台のサーバから成るサーバ群を構成し、
前記管理サーバは、
前記インターネット上にて前記サーバ群を一意に表すサーバ群用識別子と、前記インターネット上にて、前記サーバ群を構成する前記複数台のサーバのそれぞれを一意に表すサーバ用識別子と、これらのサーバ用識別子の中から目的のサーバ用識別子を選択する際に用いられる選択情報とが格納された管理テーブルと、
前記インターネットを介して、前記サーバ群用識別子を用いたアクセス要求を受付けると、前記選択情報の内容に応じて前記サーバ用識別子の何れかを選択する選択手段と、
前記選択手段が選択したサーバ用識別子を用いて前記サーバへのアクセスを行い、該アクセスで得られた情報を前記アクセス要求の要求元に送信すると共に、この情報を一時的に保持し、前記サーバ群用識別子を用いたアクセス要求を再度受付けた場合には、保持している当該情報を当該アクセス要求の要求元に送信する代理アクセス手段とを備えることを特徴とするサーバシステム。
【請求項7】請求項6において、
前記サーバ群用識別子は、前記サーバ群に与えられたホスト名であり、
前記複数のサーバ用識別子のそれぞれは、各サーバに与えられたホスト名であることを特徴とするサーバシステム。
【請求項8】請求項6または7において、
前記選択情報には、前記サーバ用識別子を選択するための選択順序が含まれており、
前記選択手段は、前記選択情報の前記選択順序にしたがって前記サーバ用識別子を選択することを特徴とするサーバシステム。
【請求項9】請求項6または7において、
前記管理サーバは、各サーバの負荷状況を判断し、その判断結果を前記選択情報の一部として前記管理テーブルに格納する負荷監視手段をさらに備え、
前記選択手段は、前記サーバ用識別子を選択する際に、前記判断結果を用いて、その時点で負荷が最も少ないサーバを選び、そのサーバのサーバ用識別子を選択することを特徴とするサーバシステム。

【請求項10】請求項6または7において、
前記各サーバには、そのサーバの負荷状況を判断して、その結果を判断情報として前記管理サーバに送信する手段が設けられており、
前記管理サーバには、前記各サーバから送信されたそれぞれの判断情報を前記選択情報の一部として前記管理テーブルに格納する手段が設けられており、
前記選択手段は、前記サーバ用識別子を選択する際に、前記判断情報を用いて、その時点で負荷が最も少ないサーバを選び、そのサーバ用識別子を選択することを特徴とするサーバシステム。
【請求項11】請求項6、7、8、9または10において、
前記サーバ群は、複数群構成されており、
前記管理テーブルには、前記各サーバ群に対して与えられたサーバ群用識別子と、前記各サーバ群を構成する前記複数台のサーバのそれぞれのサーバ用識別子と、これらのサーバ用識別子の中から目的のサーバ用識別子を選択する際に用いられる選択情報とが格納され、
前記選択手段は、前記インターネットを介して、前記サーバ群用識別子を用いたアクセス要求を受付けると、前記選択情報の内容に応じて、該サーバ群用識別子で示されるサーバ群に含まれている各サーバのサーバ識別子の中から目的のサーバ識別子を選択することを備えたことを特徴とするサーバシステム。
【発明の詳細な説明】
【0001】
【発明の属する技術分野】本発明は、インターネットを介して情報発信を行う複数台のサーバと、該複数台のサーバのそれぞれを管理する管理サーバとを備えたサーバシステムに関するものである。
【0002】
【従来の技術】インターネット上には、現在、ファイル転送のためのプロトコルであるFTP（File Transfer Protocol）を利用したFTPサーバや、テキストデータとグラフィックデータが混在したマルチメディア情報を扱うプロトコルであるHTTP（Hyper Text Transfer Protocol）を利用したWWW（World Wide Web）サーバなどが情報発信ソースとして活躍している。
【0003】インターネットは、全世界の不特定多数のユーザがアクセス可能なネットワークで、現在急速な勢いで広まりつつあるが、一方では、人気の高いサーバにユーザからのアクセスが集中し、そのサーバに関して情報発信処理速度が著しく低下するという問題がある。また、目的のサーバが常に稼働しているとは限らず、故障やメインテナンス作業等のため、そのサーバにアクセスできないということもある。
【0004】このような状況に対処するため、インターネットに接続された情報発信サーバでは、従来より、以下に述べる2つの手法がよく用いられる。
【0005】第1の手法は、一つのサーバにアクセスが集中しないよう、同一の情報を複数台のサーバのそれぞれに持たせる方法である。
【0006】具体的には、図10に示すような、FTPサーバのミラーサイトがこれに相当する。
【0007】同図において、オリジナルサイト53のFTPサーバA54は、元情報を所有しており、ミラーサイト51のFTPサーバB52は、オリジナルサイト53のFTPサーバA54から該情報を定期的に引き出し、自らに格納する。これにより、端末110を利用して情報をアクセスしようとするユーザは、FTPサーバA54が高負状態あるいは停止状態にあっても、FTPサーバB52から必要な情報を手に入れることができるようになる。また、FTPサーバA54から見れば、同サーバへのアクセスが分散される結果となり負荷が軽減される。
【0008】第2の手法は、情報発信サーバへのアクセス経路上に、情報発信サーバからユーザが引き出した情報を一時的に格納する（キャッシングする）サーバを配置し、その情報に対して、例えば、別のユーザがアクセスしようとした場合は、この情報を情報発信サーバからではなくキャッシングしているサーバから供給する方法である。
【0009】例えば、図11に示すような、WWWサーバへのアクセス経路上に配置されたプロキシサーバがこれに相当する。
【0010】同図において、プロキシサーバ62は、WWWサーバ61の情報を得ようとする、端末110からのアクセス要求を受けると、端末110に代わって、WWWサーバ61へアクセスする。その後、プロキシサーバ62は、この代理アクセスで得た情報をキャッシュデータ63として保持する共に、アクセス要求を発した端末110に該情報を返送する。その後、プロキシサーバ62は、他の端末（図示省略）から該情報に対するアクセス要求を受付けると、該WWWサーバ61にアクセスすることなく、自らが所有するキャッシュデータ63を用いて返送処理を行う。これにより、WWWサーバ61へのアクセス回数が軽減されることになる。
【0011】
【発明が解決しようとする課題】しかしながら、前述した第1の手法においては、各サーバの負荷軽減対策や、一つのサーバが停止した場合におけるバックアップ機能については実現されているものの、端末利用ユーザが必要な情報を得るためには、オリジナルサイトとミラーサイトのどちらかをユーザ側で選択しなければならない。このような選択作業は非常に煩雑であり、また、どのサーバが空いているのか等の判断をユーザに強いることになる。

【００１２】また、前述した第２の手法においては、情報源となるサーバ（図１１：ＷＷＷサーバ６１）へのアクセス回数は軽減されるものの、該サーバの停止時においては、キャッシュ用サーバ（図１１：プロキシサーバ６２）が、これにアクセスすることができず、結果として、ユーザへの情報提供ができなくなる。

【００１３】このような問題点に鑑み、本発明の目的は、端末利用ユーザに特別な操作を強いること無く、サーバ群内におけるアクセス分散と障害発生時等のバックアップが図れるインターネット用サーバシステムを提供することにある。

【００１４】

【課題を解決するための手段】上記目的を達成するための本発明の一態様によれば、インターネットを介して情報発信を行う複数台のサーバと、該複数台のサーバのそれぞれを管理する管理サーバとを備えたサーバシステムにおいて、前記複数台のサーバの少なくとも一部は、同一情報の発信を行う複数台のサーバから成るサーバ群を構成し、前記管理サーバは、前記サーバ群に対して与えられたホスト名と、前記サーバ群を構成する前記複数台のサーバのそれぞれの通信用アドレスと、これらの通信用アドレスの中から目的の通信用アドレスを選択する際に用いられる選択情報とが格納された管理テーブルと、前記インターネットを介して、前記ホスト名に対応する通信用アドレスを要求されると、前記選択情報の内容に応じて、前記管理テーブルに格納されている通信用アドレスの何れかを選択する選択手段と、前記選択手段が選択した通信用アドレスを前記通信用アドレスの要求元に送信するネームサービス手段とを備えたことを特徴とするサーバシステムが提供される。

【００１５】上記目的を達成するための本発明のその他の態様によれば、インターネットを介して情報発信を行う複数台のサーバと、該複数台のサーバのそれぞれを管理する管理サーバとを備えたサーバシステムにおいて、前記複数台のサーバの少なくとも一部は、同一情報の発信を行う複数台のサーバから成るサーバ群を構成し、前記管理サーバは、前記インターネット上にて前記サーバ群を一意に表すサーバ群用識別子と、前記インターネット上にて、前記サーバ群を構成する前記複数台のサーバのそれぞれを一意に表すサーバ用識別子と、これらのサーバ用識別子の中から目的のサーバ用識別子を選択する際に用いられる選択情報とが格納された管理テーブルと、前記インターネットを介して、前記サーバ群用識別子を用いたアクセス要求を受付けると、前記選択情報の内容に応じて前記サーバ用識別子の何れかを選択する選択手段と、前記選択手段が選択したサーバ用識別子を用いて前記サーバへのアクセスを行い、該アクセスで得られた情報を前記アクセス要求の要求元に送信すると共に、この情報を一時的に保持し、前記サーバ群用識別子を用いたアクセス要求を再度受付けた場合には、保持している当該情報を当該アクセス要求の要求元に送信する代理アクセス手段とを備えることを特徴とするサーバシステムが提供される。

【００１６】

【発明の実施の形態】以下、本発明の一実施形態について説明する。

【００１７】図１には、本実施形態のサーバシステムの全体構成が示されている。

【００１８】このサーバシステムは、インターネット１１１を介して情報発信を行う複数の情報発信サーバ群（情報発信サーバ群１０９１、１０９２、…、１０９Ｎ）と、これらの情報発信サーバ群に対して行われる端末からのアクセスを管理する管理サーバ１０１とを有する。情報発信サーバ群１０９１、１０９２、…、１０９Ｎは、それぞれ、複数台のサーバから構成される。情報発信サーバ群１０９１は、同一情報を有する情報発信サーバ１０９１１～１０９１Ｎから構成される。その他の情報発信サーバ群についても、これと同様に、そのサーバ群特有の情報をそれぞれの情報発信サーバが所有する。なお、特定の情報発信サーバ群へのアクセスが増加した場合は、同一情報を有するサーバを適宜追加すればよい。情報発信サーバ群１０９１、１０９２、…、１０９Ｎを管理するための管理情報は、ネームデータベース（ネームＤＢ）１０３に格納されている。ネームＤＢ１０３のデータ構造は、図２に示されている。

【００１９】ネームＤＢ１０３には、サーバネームフィールド２１とサーバアドレスフィールド２２の２つの格納フィールドが存在する。本実施形態では、あらかじめ、情報発信サーバ群１０９１、１０９２、…、１０９Ｎのそれぞれに対してホスト名を与えており、サーバネームフィールド２１には、これらのホスト名が格納されている。このホスト名は、言うなれば、その情報発信サーバ群の各情報発信サーバを代表するホスト名であり、以下、このホストを便宜上、代表ホスト名と呼ぶこととする。

【００２０】一方、本サーバシステムに設けられている全ての情報発信サーバは、それぞれ固有のサーバアドレスを有しており、サーバアドレスフィールド２２には、これらのサーバアドレスが情報発信サーバ群毎にまとまった形で格納されている。サーバネームフィールド２１の「－」は、その上の欄に格納されている内容と同じであることを意味している。情報発信サーバのアドレスは、インターネット１１１上で通信を行う際に使用する通信用アドレスで、具体的には、ＴＣＰ／ＩＰ通信を行うためのＩＰアドレスである。

【００２１】なお、情報発信サーバ群１０９１、１０９２、…、１０９Ｎのそれぞれに与えられる代表ホスト名は、通常のホスト名、すなわちＤＮＳ（Ｄｏｍａｉｎ Ｎａｍｅ Ｓｙｓｔｅｍ）上で一義的に定まるホスト名と何らか変るものではない。

【0022】ここで、ドメイン（Domain）について簡単に説明する。

【0023】インターネットの構築がはじまった時点では、ネットワークに接続された個々のコンピュータを識別するために、そのホスト名とIPアドレスは、所定の組織で一元管理されていた。当初は、コンピュータの数が数百台のオーダであり、すべてのコンピュータのホスト名とIPアドレスを一つのファイルに登録することも可能であった。

【0024】しかしながら、ネットワークの拡大とともに、すぐにこの方法の限界が明らかになった。すなわち、次々に行われる新規登録や更新に莫大な時間を要すると共に、管理元の作業量が増大し、結果として収拾がつかなくなってしまった。

【0025】ドメインは、このような問題を解決するために提案された概念で、世界中に存在する数多くのコンピュータを一意に名前付けするために導入された。

【0026】インターネットのドメイン空間は、階層的に管理されており、DNSは、この管理体系にマッチするように設計されている。インターネットでは、各ドメインの情報をもつDNSサーバが分散配置され、ドメイン空間と同様に階層的に管理されている。本実施形態の管理サーバも、このDNSサーバの一種と考えることができる。

【0027】図2においては、情報発信サーバ群1091を示す代表ホスト名（serv．bbb．aaa）と、情報発信サーバ群1092を示す代表ホスト名（other．bbb．aaa）が図示されている。サーバ群管理テーブル105には、これら以外の情報発信サーバ群の代表ホスト名についても全て登録されている。情報発信サーバ10911のサーバアドレス192．128．16．1をはじめ、情報発信サーバ群1091を構成する各情報発信サーバのサーバアドレスは、情報発信サーバ群1091の代表ホスト名（serv．bbb．aaa）に対応付けられた形で格納されている。情報発信サーバ群1092を構成する各情報発信サーバのサーバアドレスは、情報発信サーバ群1092の代表ホスト名（other．bbb．aaa）に対応付けられた形で格納されている。

【0028】管理サーバ101は、外部からのネームサービス要求を受けて、その応答処理を行うネームサービス部102と、ネームサービス部102にあってネームDB103から該当情報の検索・抽出を行うネームDB検索処理部106と、ネームサービス部102から、ネームDB検索処理部106によるネームDB103の検索結果であるサーバ群情報107を受け取り、サーバ群情報107が示すサーバ群からサーバ1台を選択し、その結果を選択サーバ情報108としてネームサービス部102に通知するサーバ選択部104と、サーバを選択する際にサーバ選択部104が参照するサーバ群管理テーブル105とを備える。

【0029】以上の構成を有する管理サーバ101は、情報発信サーバ群1091～109Nと同一LAN（Local Area Network）112上に存在しており、このLAN112がインターネット111に接続している。インターネット111には無数の端末が接続しており、各ユーザは、それぞれの端末を用いて本サーバシステムの各情報発信サーバ群をアクセスすることが可能である。なお、図1には、代表的に端末110が図示されている。

【0030】つぎに、端末110から情報発信サーバ群1091をアクセスする場合を例にとって、本サーバシステムの動作について説明する。ここでは、情報発信サーバ群1091に付与されている代表ホスト名をユーザが予め知っているものとして話しを進める。

【0031】ユーザは、まず、情報発信サーバ群1091の代表ホスト名（serv．bbb．aaa）を端末110に入力する。端末110は、この入力操作を受けて、代表ホスト名に対応するサーバアドレスを管理サーバ101に問い合わせる。具体的には、この際、代表ホスト名（serv．bbb．aaa）を示す情報が管理サーバ101に送られる。ここまでは、従来のDNSサーバにネームサービスを要求する場合と同じである。

【0032】管理サーバ101のネームサービス部102は、端末110からのアドレス問い合わせ、すなわち、代表ホスト名（serv．bbb．aaa）を示す情報を受け取ると、ネームDB検索処理部106を起動する。ネームDB検索処理部106は、図5に示すネームDB検索処理を実行する。

【0033】同図に示すように、ネームDB検索処理部106は、まず、端末110から送られた代表ホスト名をキーにネームDB103を検索する（ステップ51（S51））。これにより、キーにした代表ホスト名に対応する全てのサーバアドレスが抽出される。ここでは、代表ホスト名（serv．bbb．aaa）に対応するサーバアドレス（192．128．16．1、192．128．16．4、…）が抽出される。次に、ネームDB検索処理部106は、見つかったサーバアドレスの全てを先程の代表ホスト名と共にサーバ群情報107としてサーバ選択部104に通知する（S52）。サーバ選択部104は、サーバ群管理テーブル105を参照し、その内容に応じて、情報提供を行わせるに最適なサーバを一つ選択する。この選択処理の詳細は後述することとする。選ばれたサーバのホスト名とサーバアドレスは、選択サーバ情報108として、ネームサービス部102に送られる。ネームサービス部102は、このサーバアドレスを検索結果として処理し、その後、アドレス問い合わせ要求の応答処理として、このサーバアドレスを端末110に送信する（S53）。

【0034】管理サーバ101からサーバアドレスを通

知された端末１１０は、これ以後、そのサーバに対して直接、情報アクセスを行えるようになる。

【００３５】このように本実施形態によれば、或る情報発サーバ群がアクセスされる度に、その情報発信サーバ群の中から、情報発信を行うに最適なサーバが選択されるので、結果として、端末からのアクセスがその情報発信サーバ群内で分散されることになる。

【００３６】なお、特に説明しなかったが、端末１１０では、該端末１１０から送出された代表ホスト名が管理サーバ１０１に直接、あるいは、ここでは図示しないＤＮＳを介して到達するよう、あらかじめ環境設定が為されている。これについては、ＤＮＳ上で通常行われていることなので詳細は省略する。

【００３７】また、上記実施形態では、同一ＬＡＮに情報発信サーバ群と管理サーバを接続しているが、これは、情報発信サーバ群が管理サーバと同一ＬＡＮ上に存在しなくてはならないことを意味するものではない。例えば、情報発信サーバ群の各サーバはインターネット上の任意の場所に接続されていてもよい。

【００３８】つぎに、本発明のその他の実施形態について説明する。

【００３９】図３には、本実施形態のサーバシステムの全体構成が示されている。

【００４０】本サーバシステムの管理サーバ３０１は、インターネットでさまざまなマルチメディア情報の発信を行うWWW（Ｗｏｒｌｄ　Ｗｉｄｅ　Ｗｅｂ）サーバ等へのアクセスにおいて、端末からWWWサーバへのアクセスを直接受け付け、端末の代わりにWWWサーバに対する情報アクセス、及び、そのアクセス情報のキャッシングを行う、いわゆるプロキシサーバの役割を果たすものである。

【００４１】図中、１０９１～１０９Ｎは、同一情報を有するサーバ複数台からなる情報発信サーバ群である。ここでは、情報発信サーバ群１０９１が、複数台のWWWサーバから構成され、情報発信サーバ群１０９２が、複数台のＦＴＰサーバから構成されている。

【００４２】管理サーバ３０１において、３０２は、端末からの情報アクセスに応じて、情報発信サーバへの代理アクセス、アクセス情報の端末への返送、及び、そのアクセス情報の一時保持（キャッシュ処理）を行う代理アクセス部、３０６は、代理アクセス部３０２にあってサーバリスト３０３から該当情報の検索・抽出を行うサーバリスト検索処理部、３０４は、代理アクセス部３０２から、サーバリスト検索処理部３０６によるサーバリスト３０３の検索結果であるサーバ群情報３０７を受け取り、その情報が示すサーバ群の中からサーバ1台を選択し、その結果を代理アクセス部３０２に選択サーバ情報３０８として通知するサーバ選択部、３０５は、サーバ選択部３０４にあってサーバの選択処理の際に参照されるサーバ群管理テーブルである。

【００４３】サーバリスト３０３のデータ構造は、図４に示されている。

【００４４】サーバリスト３０３には、図４に示すように、代表サーバネームフィールド４１とサーバ群内サーバネームフィールド４２の２つの格納フィールドが存在する。本実施形態でも、前述と同様に、情報発信サーバ群１０９１、１０９２、…、１０９Ｎのそれぞれに対してホスト名（代表ホスト名）を与えており、サーバネームフィールド４１には、これらの代表ホスト名が格納されている。代表サーバネームフィールド４１の「－」は、その上の欄に格納されている内容と同じであることを意味している。また、サーバ群内サーバネームフィールド４２には、代表サーバネームフィールド４１に格納されている代表ホスト名で示される情報発信サーバ群を構成する各サーバのホスト名が格納されている。

【００４５】図４には、情報発信サーバ群１０９１を示す代表ホスト名（ｗｗｗ．ｂｂｂ．ａａａ）と、情報発信サーバ群１０９２を示す代表ホスト名（ｆｔｐ．ｂｂｂ．ａａａ）が図示されている。情報発信サーバ１０９１１のホスト名（ｓｅｒｖ１．ｗｗｗ－ｃ．ｂｂｂ．ａａａ）をはじめ、情報発信サーバ群１０９１を構成する各情報発信サーバのホスト名は、情報発信サーバ群１０９１の代表ホスト名（ｗｗｗ．ｂｂｂ．ａａａ）に対応付けられた形で格納されている。情報発信サーバ群１０９２を構成する各情報発信サーバのホスト名は、情報発信サーバ群１０９２の代表ホスト名（ｆｔｐ．ｂｂｂ．ａａａ）に対応付けられた形で格納されている。

【００４６】情報発信サーバ群１０９１～１０９Ｎとアクセス管理サーバ３０１は、それぞれ同一ＬＡＮ（Ｌｏｃａｌ　Ａｒｅａ　Ｎｅｔｗｏｒｋ）１１２に接続している。同ＬＡＮ１１２は、インターネット１１１に接続している。インターネット１１１には、情報発信サーバ群１０９１～１０９Ｎが発信している情報を入手する際に使用するユーザの端末１１０が接続している。

【００４７】つぎに、情報発信サーバ群１０９１の各サーバが発信している情報を端末１１０で入手する場合を例にとり、本サーバシステムの動作を説明する。ここでは、情報発信サーバ群１０９１に付与されている代表ホスト名をユーザが予め知っているものとして話しを進める。

【００４８】ユーザは、まず、情報発信サーバ群１０９１の代表ホスト名（ｗｗｗ．ｂｂｂ．ａａａ）を端末１１０に入力する。端末１１０は、この入力操作を受けて、代表ホスト名を管理サーバ３０１に送り、情報発信サーバ群１０９１へのアクセスを要求する。ここまでは、従来のプロキシサーバにアクセスを要求する場合と同じである。

【００４９】管理サーバ３０１の代理アクセス部３０２は、端末１１０から代表ホスト名（ｗｗｗ．ｂｂｂ．ａａａ）を受け取ると、サーバリスト検索処理部３０６を

起動する。サーバリスト検索処理部３０６は、図6に示すサーバリスト検索処理を実行する。

【００５０】サーバリスト検索処理部３０６は、まず、端末１１０から送られた代表ホスト名をキーにサーバリスト３０３を検索する（ステップ６１（Ｓ６１））。これにより、キーにした代表ホスト名に対応する全てのホスト名が抽出される。ここでは、代表ホスト名（ｓｅｒｖ．ｂｂｂ．ａａａ）に対応するホスト名（ｓｅｒｖ１．ｗｗｗ－ｃ．ｂｂｂ．ａａａ、ｓｅｒｖ４．ｗｗｗ－ｃ．ｂｂｂ．ａａａ、…）が抽出される。次に、サーバリスト検索処理部３０６は、見つかったホスト名全てを先程の代表ホスト名と共にサーバ群情報３０７としてサーバ選択部３０４に通知する（Ｓ６２）。サーバ選択部３０４は、サーバ群管理テーブル３０５を参照し、その内容に応じて、情報提供を行わせるに最適なサーバを一つ選択する。この選択処理の詳細は後述することとする。選ばれたサーバのホスト名は、前記代表ホスト名と共に選択サーバ情報３０８として、代理アクセス部３０２に送られる。代理アクセス部３０２は、このホスト名を用いて、そのサーバにアクセスする。代理アクセス部３０２は、この代理アクセスによって得られた情報を該管理サーバ３０１内のメモリにキャッシュデータとして格納し、さらに、このキャッシュデータを端末１１０に返送する。次回、代表ホスト名（ｗｗｗ．ｂｂｂ．ａａａ）によるアクセス要求を受け付けた場合には、このキャッシュデータが当該アクセス要求元に送信される。

【００５１】以上の動作により、端末１１０は、管理サーバ３０１を介して間接的に情報発信サーバにアクセスする。

【００５２】このように本実施形態によれば、管理サーバ３０１が行う代理アクセスが、その時点で最適なサーバに対して行われるため、結果として、そのサーバ群内のアクセスが分散される。

【００５３】以上、本発明のサーバシステムの２つの実施形態について、その全体構成と動作を中心に説明したが、続いて、管理サーバの備えるサーバ選択部の詳細を説明する。

【００５４】ここでは、便宜上、図1に示したサーバシステムを例にとって話しを進める。

【００５５】まず、図7に示したサーバ群管理テーブル１０５を用いた場合のサーバ選択部１０４の動作について説明する。

【００５６】ネームサービス１０２からサーバ選択部１０４に送られるサーバ群情報１０７には、ネームＤＢ検索処理（図5参照）でキーとした代表ホスト名と、該代表ホスト名をキーとして検索された検索結果である複数のサーバアドレスが含まれているが、このうち、図7のサーバネームフィールド１０５１には、前記代表ホスト名が格納される。また、サーバアドレスフィールド１０５２には、検索結果である前述の複数のサーバアドレスが格納される。

【００５７】また、サーバ選択部１０４が選択したサーバのホスト名は、選択サーバ情報１０８としてネームサービス１０２に送られるが、選択サーバ情報１０８で通知したサーバアドレスについては、アクセスフラグフィールド１０５３に「１」（１：通知）が設定される。なお、その他のサーバアドレスについては、「０」（０：未通知）を設定される。

【００５８】そして、サーバ選択部１０４は、サーバ群情報１０７を受け取ると、サーバ群情報１０７の示す代表ホスト名がサーバネームフィールド１０５１に存在するか否かを判断し、該代表ホスト名がサーバネームフィールド１０５１に無ければ、この代表ホスト名と共に、これに対応付けられている全てのサーバアドレス（サーバ群情報１０７に含まれている全てのサーバアドレス）をサーバ群管理テーブル１０５に登録する。このとき、一つのサーバアドレスについては、そのアクセスフラグフィールド１０５３に”１”を設定する。他のアクセスフラグフィールド１０５３には、”０”を設定する。同図においては、サーバアドレス１９２．１２８．１６．１のアクセスフラグフィールド１０５３に”１”を設定し、これ以外は”０”を設定している。

【００５９】一方、すでに同一代表ホスト名がサーバネームフィールド１０５１に格納されている場合は、つぎの作業を行う。

【００６０】例えば、サーバ群管理テーブル１０５の内容が図7の状態に既に設定されている場合、サーバアドレス１９２．１２８．１６．４のアクセスフラグフィールド１０５３に”１”を設定すると共に、サーバアドレス１９２．１２８．１６．１のアクセスフラグフィールド１０５３に”０”を設定する。なお、最後尾に位置する、サーバアドレス１９２．１２８．１６．１２のアクセスフラグフィールド１０５３に”１”が設定されていた場合は、先頭行に位置する、サーバアドレス１９２．１２８．１６．１のアクセスフラグフィールド１０５３に”１”を設定すると共に、サーバアドレス１９２．１２８．１６．１２のアクセスフラグフィールド１０５３に”０”を設定する。

【００６１】以上の処理を行った場合、設定値である”１”は、アクセスフラグフィールド１０５３を順番に移動していくかのように見える。

【００６２】つぎに、サーバ選択部１０４は、アクセスフラグフィールド１０５３内に”１”を設定した行のサーバアドレスフィールド１０５２内に格納されているサーバアドレスを選択サーバ情報１０８としてネームサービス部１０２に通知する。

【００６３】このように情報発信を行わせるサーバを次々に選択していけば、端末からのアクセスは、その情報発信サーバ群内の各サーバに対してほぼ均等に分散されるようになる。

【0064】つぎに、図8に示したサーバ群管理テーブル105を用いた場合のサーバ選択部104の動作について説明する。

【0065】ここでは、情報発信サーバ群を構成する各サーバの負荷に応じて、アクセスするサーバを選択する。

【0066】図8において、同じ構成要素については、同じ番号を付して説明を省略する。

【0067】このサーバ群管理テーブル105は、先程のアクセスフラグフィールド1053に代えて、負荷レベルフィールド1054を備えている点に特徴がある。負荷レベルフィールド1054には、各サーバの負荷状況を示す情報が格納される。

【0068】また、選択部104では、新たなにサーバ負荷監視部113が設けられている。サーバ負荷監視部113は、サーバ群管理テーブル105に登録されているサーバ（すなわち、アドレスフィールド1052にサーバアドレスが格納されているサーバ）の負荷状況を定期的に監視し、負荷レベルフィールド1054にその負荷レベルを設定する。負荷レベルは予め数段階（ここでは、負荷レベルA、B、Cの3段階）に分けられている。具体的には、サーバ負荷監視部113は、サーバアドレスフィールド1052に格納されているサーバアドレスのサーバに対して定期的に監視コマンドを送信し、その応答時間を集計し、その結果を、負荷レベルA／B／Cの何れかに分けて負荷レベルフィールド1054に設定する。ここで、上記監視コマンドに対して何も応答しないサーバについては、その負荷レベルフィールド1053に対して何も設定しない。サーバが何も応答しない場合、そのサーバは、保守作業中、あるいは、障害の発生による停止状態にあることが予想される。

【0069】そして、サーバ選択部104は、サーバ群情報107を受けた時点で、サーバ群管理テーブル105に登録されているサーバの中から最も負荷の軽いものを選択し、このサーバのアドレス等を選択サーバ情報108としてネームサービス部102に通知する。この際、負荷レベルフィールド1053に何も設定されていないサーバについては、あらかじめ選択枝から除外しておくこととする。

【0070】このような処理を行えば、端末からのアクセスは、目的の情報発信サーバ群の中から、その時点で最も負荷の軽いサーバに対して行われるようになる。また、稼働していないサーバへのアクセスも禁止されることになる。

【0071】なお、各サーバの負荷に応じてアクセスするサーバを選択する場合、サーバシテムは図9に示すように構成しても構わない。

【0072】同図において、各情報発信サーバには、そのサーバの負荷をモニターし、その負荷状況を管理サーバ101に報告する負荷モニタ115が設けられている。管理サーバ101は、各サーバからの負荷状況報告を収集し、その結果を負荷レベルとしてサーバ群管理テーブル105に格納する負荷状況収集部114を備える。ここで、負荷モニタ部115から報告のないサーバについては負荷レベルを設定しない。

【0073】そして、サーバ選択部104は、サーバ群情報107を受け付けると、サーバ群管理テーブル105に登録されているサーバの中から負荷レベル情報が設定されているものを探し出し、その中から最も負荷の軽いものを選択し、このサーバのサーバアドレスを選択サーバ情報108としてネームサービス部102に通知する。

【0074】このように各サーバの負荷状況を、そのサーバ自身にモニターさせれば、より正確な負荷情報を得ることができるようになる。

【0075】以上、管理サーバの備えるサーバ選択部の各種実施形態について説明したが、これらは、図3に示したサーバシステムにも適用可能である。

【0076】

【発明の効果】本発明によれば、障害発生時などのバックアップ対策として、また、ユーザからのアクセスを分散させる目的で、同一の情報を有する複数台のサーバを設けた場合であっても、情報発信を行わせるに最適なサーバが自動的に選択されるため、ユーザは、複数台のサーバの存在を意識することなく、必要な情報を手に入れることができるようになる。

【0077】また、本発明によれば、情報発信を行わせるに最適なサーバは、各サーバの負荷状況に応じて、また、稼働していないサーバは対象外として選択されるため、ユーザは、常に最良の環境でアクセスを行うことができるようになる。

【0078】その上、アクセス負荷が増加しても、本発明を提供しておくことにより、同一情報を有するサーバを追加するだけで、アクセス負荷に応じた性能補強が可能となる。

【図面の簡単な説明】

【図1】本発明に係るサーバシステムの一実施形態の全体構成図。

【図2】図1に示した管理サーバが備えるネームDBの構造を示した説明図。

【図3】本発明に係るサーバシステムのその他の実施形態の全体構成図。

【図4】図3に示した管理サーバが備えるサーバリストの構造を示した説明図。

【図5】図1に示した管理サーバが行うネームDB検索処理の一例を示したフローチャート。

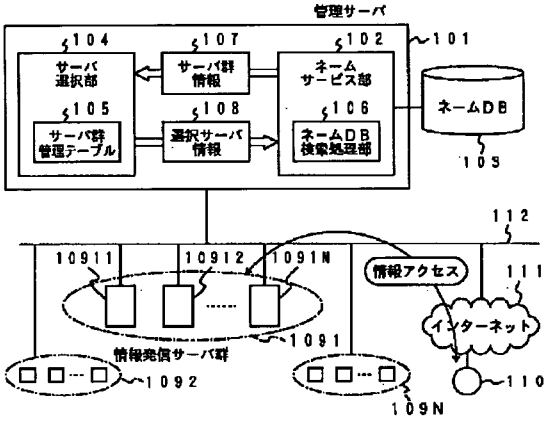
【図6】図3に示した管理サーバが行うサーバリスト検索処理の一例を示したフローチャート。

【図7】図1に示した管理サーバの選択部の一例（その1）を示した説明図。

【図8】図1に示した管理サーバの選択部の一例（その2）を示した説明図。
【図9】本発明に係るサーバシステムのさらに別の実施形態の全体構成図。
【図10】従来のサーバシステム（FTPミラー）の一例を示した説明図。
【図11】従来のサーバシステム（プロキシアクセス）一例を示した説明図。
【符号の説明】
101、301…管理サーバ、
102…ネームサービス部、
103…ネームDB、
104、304…サーバ選択部、
105、305…サーバ群管理テーブル、
106…ネームDB検索処理部、
107、307…サーバ群選択情報、
108、308…選択サーバ情報、
1091～109N…情報発信サーバ群、
10911～1091N…情報発信サーバ、
110…端末、
111…インターネット、
112…LAN（Local Area Network）、
302…代理アクセス部、
303…サーバリスト、
306…サーバリスト検索処理部、
113…サーバ負荷監視部、
114…負荷状況収集部、
115…負荷モニタ部

【図1】

図 1



【図2】

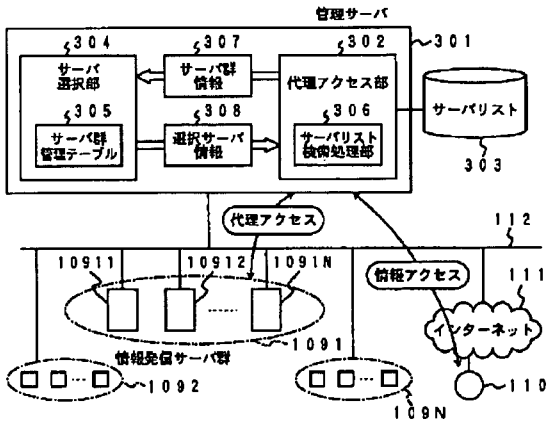
図 2

21	22
サーバネーム	サーバアドレス
serv.bbb.aaa	192.128.16.1
−	192.128.16.4
−	⋮
−	192.128.16.12
other.bbb.aaa	192.128.64.18
⋮	⋮

103

【図3】

図 3



【図4】

図 4

41	42
代表サーバネーム	サーバ群内サーバネーム
www.bbb.aaa	serv1.www-c.bbb.aaa
−	serv4.www-c.bbb.aaa
−	⋮
ftp.bbb.aaa	serv2.ftp-c.bbb.aaa
−	serv1.ftp-c.bbb.aaa
−	⋮

303

【図7】

図 7

104

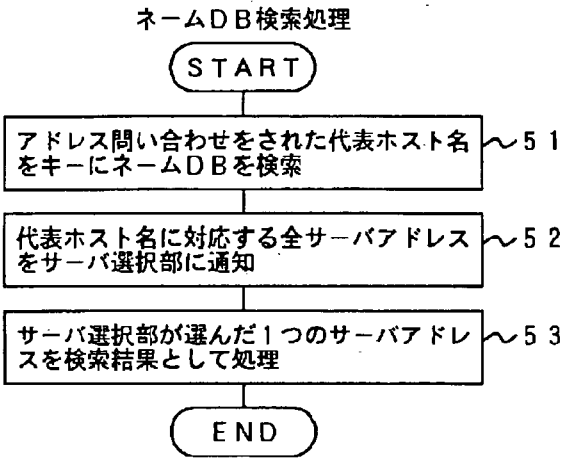
105
サーバ群管理テーブル

1051	1052	
サーバネーム	サーバアドレス	アクセスフラグ
serv.bbb.aaa	192.128.16.1	1
−	192.128.16.4	0
−	⋮	⋮
−	192.128.16.12	0

1053

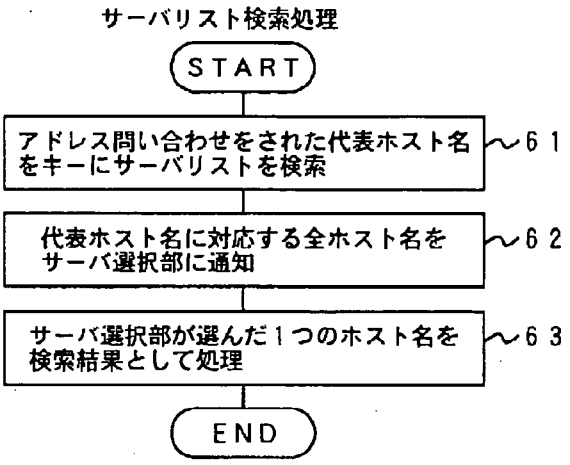
【図5】

図 5



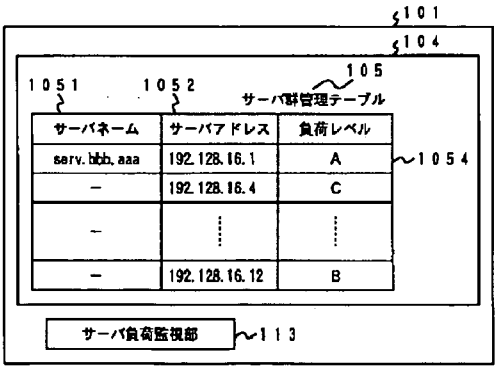
【図6】

図 6



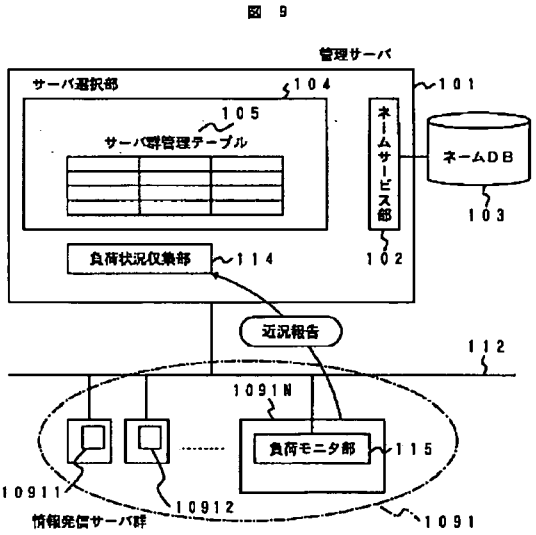
【図8】

図 8



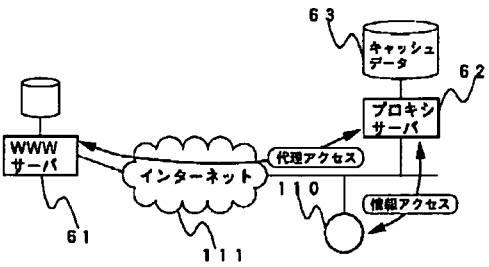
【図9】

図 9



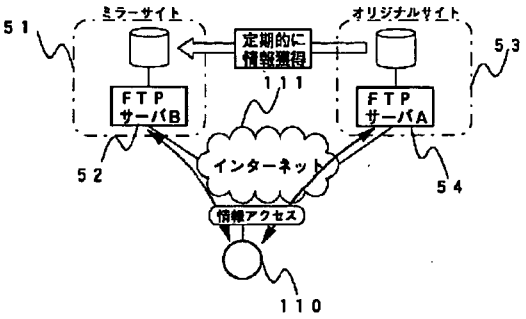
【図11】

図11



【図10】

図10



フロントページの続き

(72)発明者　綿貫　達哉
神奈川県川崎市幸区鹿島田890番地　株式会社日立製作所情報・通信開発本部内

(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平10－93552
(43)公開日　平成10年(1998)４月10日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｈ０４Ｌ　12/00		Ｈ０４Ｌ　11/00	
Ｇ０６Ｆ　13/00	３５５	Ｇ０６Ｆ　13/00	３５５

審査請求　未請求　請求項の数５　ＯＬ　（全７頁）

(21)出願番号　特願平8－240114

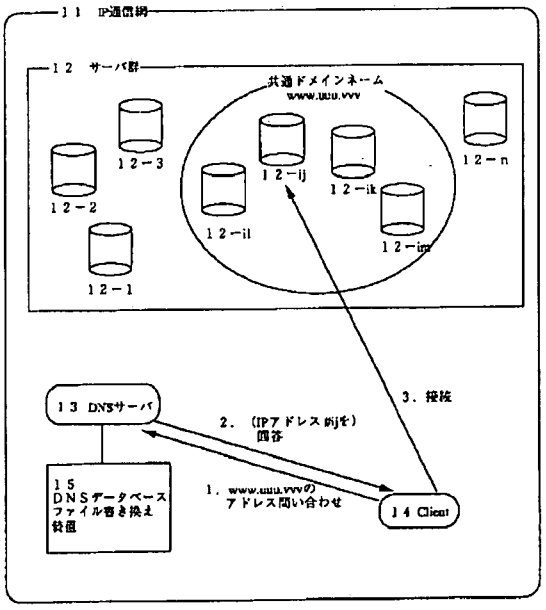
(22)出願日　平成８年(1996)９月11日

(71)出願人　000004226
日本電信電話株式会社
東京都新宿区西新宿三丁目19番２号
(72)発明者　村上　英世
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内
(72)発明者　原　博之
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内
(72)発明者　中島　伊佐美
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内
(74)代理人　弁理士　磯村　雅俊　（外１名）

最終頁に続く

(54)【発明の名称】　共通識別子を持つ複数ホストとの通信接続方法

(57)【要約】
【課題】故障または過負荷のホストを他のホストに分散して、分散元ホストの負荷を軽減することができ、継続的サービス品質を維持する。また、発信側ホストに意識させずに、通信相手を変更できるようにする。
【解決手段】共通識別子をサーバ群１２に付与し、それ以外のホスト１４がサーバ群１２と通信する場合には、共通識別子（共通ドメインネーム）を用いてアクセスすると、相手（１２－ｉ１）が故障または過負荷の場合には、接続先変更アルゴリズム１５に基づいてサーバ群１２内の他のホスト（１２－ｉｊ）に接続先を変更することにより、変更を意識させずに発信できるようにする。



【特許請求の範囲】
【請求項１】通信網を介して接続された複数のホストが、相互間で通信するための通信網の通信接続方法において、
複数の場所に設置され、物理的に異なる複数のホストからなるホスト群に対して共通の識別子を付与し、
該通信網内に、上記共通識別子から該共通識別子とは異なる各ホスト固有の識別子に変換する変換サーバを設置し、
上記ホスト群に含まれない任意のホストから、該ホスト群に含まれる任意のホストに対して発信する場合、発信側のホストは上記共通識別子を用いて上記変換サーバに問い合わせを行い、
該変換サーバは接続に必要な着信側ホストの固有識別子を発信側のホストに返送し、
該発信側のホストは返送された固有識別子を基にホスト群中の特定のホストに接続を実行することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【請求項２】請求項１に記載の共通識別子を持つ複数ホストとの通信接続方法において、
前記変換サーバは、共通識別子と、該共通識別子に対応するホスト群中の特定のホストの固有識別子とを対応づける対応テーブルを保持し、
該対応テーブルを予め決められたアルゴリズムに従って書き換えることにより、接続先の特定のホストを変更することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【請求項３】請求項２に記載の共通識別子を持つ複数ホストとの通信接続方法において、
前記対応テーブルの書き換えアルゴリズムとして、累積接続数が予め設定しておいた特定の接続数に達すると、予めホストに割り振られた順番通りに次のホストに順次接続先を変更することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【請求項４】請求項２に記載の共通識別子を持つ複数ホストとの通信接続方法において、
前記対応テーブルの書き換えアルゴリズムとして、予め設定しておいた時間間隔毎に、予めホストに割り振られた順番通りに次のホストに順次接続先を変更することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【請求項５】請求項２に記載の共通識別子を持つ複数ホストとの通信接続方法において、
前記対応テーブルの書き換えアルゴリズムとして、対応付けられたホスト群中の特定のホストの負荷が予め設定しておいた閾値を超えた場合、あるいは該特定のホストに故障が発生した場合に、該ホストに代るホストとして正常に作動しており、かつ稼働負荷が上記閾値を超えていない別のホストを選択して、該選択されたホストに接続先を変更することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【発明の詳細な説明】
【０００１】
【発明の属する技術分野】本発明は、共通の識別子を複数のホストが共有することにより、ホストに過負荷または故障が生じたときでも、そのホストに対する接続要求を複数の他の複製ホストに分配することにより、負荷を分散させることが可能であり、かつ通信相手が変更されても発信側ホストはそれを意識せずに発信することが可能な共通識別子を持つ複数ホストとの通信接続方法に関する。
【０００２】
【従来の技術】従来、通信網を介して相互接続されている複数のホストは、公開されている着信側ホストの識別子（ネーム）を用いて接続を実行するために必要な識別子（アドレス）を変換サーバに問い合わせると、変換サーバは公開されている識別子（ネーム）と通信に必要な識別子（アドレス）との１対１の対応関係を記述している対応テーブルを参照して、通信に必要な識別子（アドレス）を発信側ホストに返送していた。発信側ホストは、返送された通信に必要な識別子（アドレス）を用いて、着信側ホストとの接続を実行している。しかしながら、ある特定のホストに接続を要求する他のホストが多数存在する場合には、そのホストへの通信が輻輳してしまい、接続が実行できない。
【０００３】
【発明が解決しようとする課題】このように、従来の複数ホスト相互間の通信方法では、ある特定のホストに接続を希望するホストが多い場合、その特定のホストあるいはその特定のホストに接続する通信路の負荷が大きくなり、接続要求がそれらの処理能力を超えてしまった時点で、そのホストへの通信が困難となる。その場合には、そのホストを示す識別子を用いてアクセスを試みる他のホストからの通信を、そのホストが受け付けられなくなる可能性も生じてくる。そこで、本発明の目的は、このような従来の課題を解決し、故障あるいは過負荷となったホストの負荷を分散させることが可能であり、多大な設備投資を省略しながら、継続的かつ良好なサービス品質を維持することが可能な共通識別子を持つ複数ホストとの通信接続方法を提供することにある。また、本発明の他の目的は、通信相手の識別子が変更になっても、発信側ホストにそれを意識させることなく発信させることが可能な共通識別子を持つ複数ホストとの通信接続方法を提供することにある。
【０００４】
【課題を解決するための手段】上記目的を達成するため、本発明の共通識別子を持つ複数ホストとの通信接続方法では、共通識別子を複数のホストからなるホスト群に付与し、該ホスト群に含まれないホストは該ホスト群に含まれるホストと通信する際には、共通識別子から固

有識別子に変換する機能を備えた変換サーバに問い合わせることにより、固有識別子を用いて接続を実行する。ホスト群中の特定のホストが故障や過負荷の場合、あるいは通信網の故障や輻輳が原因で通信が困難な状況が生じた場合でも、該ホスト群に含まれないホストに意識させることなく、変換サーバがホスト群内の他のホストに接続先を変更することにより、継続的に通信サービスを提供することができるようにした。また、正常に稼働している場合においても、ホスト群内でホスト群に含まれないホストとの接続を分散させることにより、負荷を分散させることが可能になった。そのために、（ａ）共通識別子を基に、通信網内で共通識別子を受けて通信に必要な各ホスト毎に異なる固有の識別子を、共通識別子とホスト固有の識別子に対応関係を把握している変換サーバに対して問い合わせ、変換サーバで共通識別子に該当する複数のホストの中から１つのホストに固有の識別子を選択し、これを問い合わせたホストに通知する。

（ｂ）変換サーバは、同一の共通識別子が付与されたホストの稼働状況、ホストに接続する通信網の通信状況を監視することにより、ホストに加わる負荷、ホストの故障、あるいはホストに接続する通信網の通信量が特定の閾値を超えたことを検出したとき、上記情報に基づいて複数のホストの中から適切なホストを選択し、固有の識別子を割り当てる。

【０００５】

【発明の実施の形態】以下、本発明の実施例を、図面により詳細に説明する。

（第１実施例）図１は、本発明の第１の実施例を示す複数ホスト相互間の通信接続方法の説明図であり、図２は図１における共通識別子と固有アドレスとの対応表および書き換え装置を示す図である。図１では、共通識別子としてドメインネームを、また固有識別子としてＩＰアドレスをそれぞれ用い、かつ複数のサーバを用いた場合の接続方法を示している。また図２では、ＤＮＳ（ドメインネームシステム）サーバおよびＤＮＳデータベースファイル書き換え装置の機能構成を示している。この場合、ドメイン・ネームを用いるインターネットアクセスサービスであれば、どれでも適用することができるが、以下ではWWW（World Wide Web)を例として説明する。図１において、１１はＩＰルータにより構成されるＩＰ通信網であり、１２は１２－１，１２－２，・・・，１２－ｎのｎ台のWWWサーバからなるサーバ群、１３はドメイン・ネーム・システム・サーバ（ＤＮＳサーバ）、１４はWWWブラウザを備えたクライアント・ホストである。円で囲まれているサーバ群１２の一部は、ＨＴＭＬファイルａａａ．ｈｔｍｌのコピーを蓄積するサーバ１２－ｉ１，・・・，ｉｍ（１≦ｍ≦ｎ）がそれぞれ異なるＩＰアドレス＃ｉ１，・・・，＃ｉｍに対応しており、かつドメイン・ネームｗｗｗ．ｕｕｕ．ｖｖｖを共有する。

【０００６】図１において、クライアント・ホスト１４は、WWWブラウザを用いてａａａ．ｈｔｍｌを閲覧するために、予め公開されているＵＲＬ（ｈｔｔｐ：／／ｗｗｗ．ｕｕｕ．ｖｖｖ／ａａａ．ｈｔｍｌ）を用いてアクセスを実行する。その際に、クライアント・ホスト１４は、ドメイン・ネームｗｗｗ．ｕｕｕ．ｖｖｖに該当するサーバのＩＰアドレスを入手するために、ＤＮＳサーバ１３に問い合わせを行う。図２において、この時、ＤＮＳサーバ１３が持つＤＮＳデータベースファイル１３－１において、ドメイン・ネームｗｗｗ．ｕｕｕ．ｖｖｖにＩＰアドレス＃ｉｊ（１≦ｊ≦ｍ）が対応している場合、ＤＮＳサーバ１３はクライアント・ホスト１４にＩＰアドレス＃ｉｊを通知する。クライアント・ホスト１４は、それを受け取ると、ＩＰアドレス＃ｉｊを用いてサーバ１２－ｉｊにｈｔｔｐ　ｒｅｑｕｅｓｔを送り、サーバ１２－ｉｊからａａａ．ｈｔｍｌをダウンロードする。ＤＮＳデータベースファイル書き換え装置１５は、図２に示すように、ＤＮＳデータベースファイル書き換えプログラム１５－１と、そのプログラム１５－１により参照される共通ドメインネーム／サーバＩＰアドレス対応テーブルファイル１５－２から構成される。予め設定された接続先変更アルゴリズムに基づいてＤＮＳデータベースファイル書き換えプログラム１５－１が作動し、ＤＮＳデータベースファイル１３－１のドメイン・ネームｗｗｗ．ｕｕｕ．ｖｖｖに対応するＩＰアドレスを＃ｉｊから＃ｉｋ（１≦ｋ≦ｍ）に変更する。この際に、ＤＮＳデータベースファイル書き換えプログラム１５－１は、共通ドメイン・ネーム／ＩＰアドレス対応テーブル１５－２を参照する。共通ドメイン・ネーム／ＩＰアドレス対応テーブル１５－２には、共通ドメイン・ネームｗｗｗ．ｕｕｕ．ｖｖｖと、それを用いているサーバ１２－ｉ１，・・・，１２－ｉｍのＩＰアドレス＃ｉ１，・・・，＃ｉｍとの対応が全て記憶されている。

【０００７】ＤＮＳデータベースファイル１３－１の変更後は、クライアント・ホスト１４がａａａ．ｈｔｍｌをダウンロードするために、ドメイン・ネームｗｗｗ．ｕｕｕ．ｖｖｖに該当するＩＰアドレスをＤＮＳサーバ１３に問い合わせると、その回答としてＩＰアドレス＃ｉｋが通知されるので、クライアント・ホスト１４はサーバ１２－ｉｋにアクセスすることになる。接続先変更アルゴリズムの例としては、次のような手順で行われる。

（１）累積接続数が予め設定しておいた特定の接続数に達すると、１２－ｉ１，１２－ｉ２，・・・，１２－ｉｍ，１２－ｉ１，・・・のような順番に接続するように、ＤＮＳデータベースファイル１３－１を書き換える。

（２）予め設定しておいた時間間隔毎に、１２－ｉ１，１２－ｉ２，・・・，１２－ｉｍ，１２－ｉ１，・・・

のような順番に接続するように、ＤＮＳデータベースファイル１３－１を書き換える。

（３）サーバ１２－ｉｊの負荷が予め設定しておいた閾値を超えた場合、あるいはサーバ１２－ｉｊに故障が発生した場合には、サーバ１２－ｉｊからそれらの異常の発生の通知をＤＮＳデータベースファイル書き換え装置１５が受けて、サーバ１２－ｉｊに代わるサーバとして、正常に作動しており、かつ稼働負荷が閾値を超えていないサーバ１２－ｉｋを選択し、ＤＮＳデータベースファイル１３－１をサーバ１２－ｉｋに書き換える。

【０００８】（第２実施例）図３は、本発明の第２の実施例を示す複数ホストとの通信接続方法の説明図であり、図４は、図３における共通識別子と固有アドレスとの対応表および書き換え装置の構成図である。ここでは、本発明をＡＴＭ（Asynchronous Transfer Mode)上でＩＰルーチングを実現する通信網に適用した場合のネットワークを示している。そして、共通識別子として共通のＩＰアドレスを、それぞれ異なるＡＴＭアドレスを持つホストが用いる場合を説明する。図３において、２１はＡＴＭスイッチおよびＩＰルータにより構成されるＩＰｏｖｅｒ　ＡＴＭ通信網、２２は２２－１，２２－２，・・・，２２－ｎのｎ台のホストからなるホスト群、２３はＡＴＭ　ＡＲＰ（Address Resolution Protocol)サーバ、２４はＡＴＭ　ＡＲＰサーバ２３のアドレス変換テーブルを書き換えるアドレス変換テーブル書き換え装置、２５はホスト群２２に含まれないホストである。円で囲まれているホスト群は、ホスト群２２の一部であって、あるファイルＦのコピーを蓄積するホスト２２－ｉ１，・・・・２２－ｉｍ（１≦ｍ≦ｎ）がそれぞれ異なるＡＴＭアドレス＃ｉ１・・・＃ｉｍに対応し、かつ共通ＩＰアドレスＩＰｃを共有している。

【０００９】図３において、ホスト２５は、ｆｔｐを用いてファイルＦを保持するホストにアクセスを試みる。その際に、通信先のホストとの間にＳＶＣ（Switched Virtual Connection)を開いてＡＴＭ通信を行うために必要な通信先のＡＴＭアドレスを得るために、ＡＴＭ　ＡＲＰサーバ２３にＩＰアドレスＩＰｃに該当するホストのＡＴＭアドレスの問い合わせを行う。このとき、ＡＴＭ　ＡＲＰサーバ２３が持つアドレス変換テーブル２３－１において、ＩＰアドレスＩＰｃにＡＴＭアドレス＃ｉｊ（１≦ｊ≦ｍ）が対応している場合、ＡＴＭ　ＡＲＰサーバ２３はホスト２５にＡＴＭアドレス＃ｉｊを通知し、それを受け取ったホスト２５はＡＴＭアドレス＃ｉｊを用いてホスト２２－ｉｊとの間にＳＶＣを張り、ファイルＦをホスト２２－ｉｊよりダウンロードする。図４において、アドレス変換テーブル書き換え装置２４は、予め設定された接続先変更アルゴリズムに基づいてアドレス変換テーブル書き換えプログラム２４－１が作動し、アドレス変換テーブル２３－１のＩＰアドレスＩＰｃに対応するＡＴＭアドレスを＃ｉｊから＃ｉｋ（１≦ｋ≦ｍ）に変更する。この際に、アドレス変換テーブル書き換えプログラム２４－１は、共通ＩＰアドレス／ＡＴＭアドレス対応テーブル２４－２を参照する。共通ＩＰアドレス／ＡＴＭアドレス対応テーブル２４－２には、共通ＩＰアドレスＩＰｃと、それを用いているホスト２２－ｉ１，・・・，２２－ｉｍのＡＴＭアドレス＃ｉｊ，・・・・，＃ｉｍとの対応が全て記憶されている。アドレス変換テーブル２３－１の変更後は、ホスト２５がＩＰアドレスＩＰｃに該当するホストと通信を行うために、そのホストのＡＴＭアドレスをＡＴＭ　ＡＲＰサーバ２３に問い合わせると、回答としてＡＴＭアドレス＃ｉｋが通知されるので、ホスト２５はホスト２２－ｉｋと通信を行うことになる。なお、接続先変更アルゴリズムとしては、第１実施例で述べた（１）～（３）と同じである。

【００１０】（第３実施例）図５は、本発明の第３の実施例を示す複数ホストとの通信接続方法の説明図であり、図６は、図５における共通識別子と固有アドレスとの対応表および書き換え装置の構成図であり、図７は、図５におけるゲートウェイサーバからクライアント・ホストに返送されるＨＴＭＬファイルの例を示す図である。ここでは、本発明をインターネットプロトコルを用いた複数のルータで構成されるＩＰ通信網におけるＷＷＷ（World Wide Web)サービスの提供に適用する場合の接続方法を示しており、かつ共通識別子として共通のドメイン・ネームを、複数のＷＷＷサーバからなるサーバ群を代表するゲートウェイ・サーバ（ＧＷ）が用いる場合を示している。図５において、３１はＩＰルータにより構成されるＩＰ通信網、３２は３２－１，３２－２，・・・，３２－ｎのｎ台のＷＷＷサーバからなるサーバ群、３３はサーバ群３２への接続要求を受け付けるゲートウェイ・サーバ（ＧＷ）、３４はＷＷＷブラウザを備えたクライアント・ホストである。円で囲まれているサーバ群は、サーバ群３２の一部であり、かつＨＴＭＬファイルｘ．ｈｔｍｌのコピーを蓄積するサーバ３２－ｉ１，・・・，３２－ｉｍ（１≦ｍ≦ｎ）がそれぞれ異なるＩＰアドレス＃ｉ１，・・・，＃ｉｍに対応する。

【００１１】クライアント・ホスト３４は、ＷＷＷブラウザを用いてｘ．ｈｔｍｌを閲覧するために、初めに予め公開されているゲートウェイ・サーバ３３のドメイン・ネームｗｗｗ．ｐｐｐ．ｑｑｑを用いたＵＲＬ（ｈｔｔｐ：／／ｗｗｗ．ｐｐｐ．ｑｑｑ／ｘ．ｈｔｍｌ）を用いてゲートウェイ・サーバ３３にアクセスする。ゲートウェイ・サーバ３３は、図６に示すように、対応テーブル３３－１とＨＴＭＬファイル生成プログラム３３－３と接続先選択プログラム３３－２とから構成されている。対応テーブル３３－１には、ＨＴＭＬファイルｘ．ｈｔｍｌとそれを蓄積するサーバのＩＰアドレス＃ｉ１，・・・，＃ｉｍとの対応が記述されている。接続先

選択プログラム３３－２は、対応テーブル３３－１を参照して予め設定されている接続先選択アルゴリズムに基づいてサーバ３２－ｉｊ（１≦ｊ≦ｍ）のＩＰアドレス＃ｉｊを選択する。ＨＴＭＬファイル生成プログラム３３－３は、この選択を基に図７に示すようなＨＴＭＬファイル３５を作成し、それをクライアント・ホスト３４に返送する。クライアント・ホスト３４は、ＨＴＭＬファイル３５を受け取ると、それに書き込まれているＩＰアドレス＃ｉｊを用いたＵＲＬ（ｈｔｔｐ：／／＃ｉｊ／ｘ．ｈｔｍｌ）が書き込まれたｈｔｍｌファイルを受け取り、それを用いてサーバ３２－ｉｋへのアクセスを実行する。なお、接続先変更アルゴリズムとしては、前述の第１実施例で述べた例と同じである。

【００１２】

【発明の効果】以上説明したように、本発明によれば、（ａ）分散元ホストに故障あるいは過負荷が生じたときには、他のホストに分散元ホストの複製を配置して、分散元ホストに対する接続要求を複数の複製ホストに分配することにより、分散元ホストの負荷を分散させることができるので、予め予測困難な過負荷や故障のために分散元ホスト所有者が多大な設備投資をすることなく、継続的かつ良好なサービス品質を維持することができる。また（ｂ）共通の識別子を複数のホストで共有することにより、通信相手の変更を発信側ホストに意識させることなく実行することができる。

【図面の簡単な説明】

【図１】本発明の第１の実施例を示す複数ホストとの通信接続方法の説明図である。

【図２】図１における共通識別子と固有アドレスとの対応表および書き換え装置の構成図である。

【図３】本発明の第２の実施例を示す複数ホストとの通信接続方法の説明図である。

【図４】図３における共通識別子と固有アドレスとの対応表および書き換え装置の構成図である。

【図５】本発明の第３の実施例を示す複数ホストとの通信接続方法の説明図である。

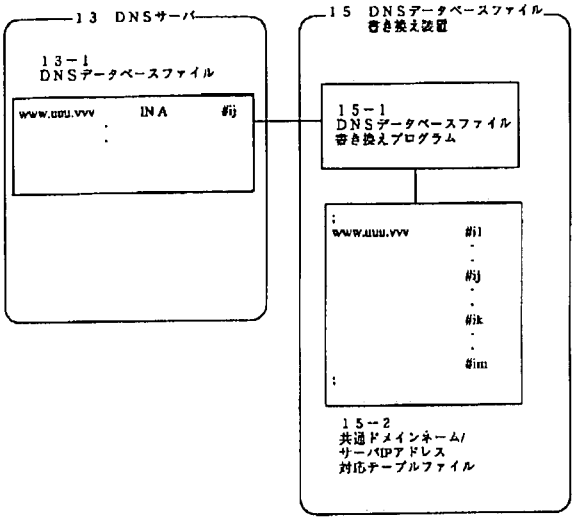
【図６】図５における共通識別子と固有アドレスとの対応表および書き換え装置の構成図である。

【図７】図５におけるゲートウェイサーバからクライアント・ホストに返送されるＨＴＭＬファイルの一例を示す図である。

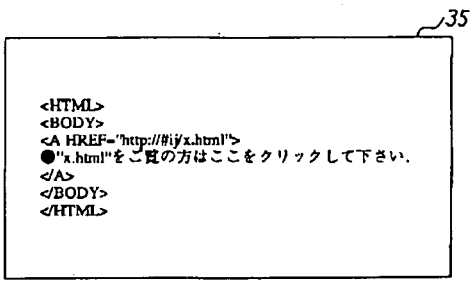
【符号の説明】

１１…ＩＰ通信網、１２…サーバ群、１２－１～１２－Ｎ…サーバ、１３…ＤＮＳサーバ、１４…クライアント・ホスト、１５…ＤＮＳデータベースファイル書き換え装置、１３－１…ＤＮＳデータベースファイル、１５－１…ＤＮＳデータベースファイル書き換えプログラム、１５－２…共通ドメインネーム／サーバＩＰアドレス対応テーブルファイル、２１…ＩＰｏｖｅｒＡＴＭ通信網、２２…ホスト群、２２－１～２２－ｎ…ホスト、２３…ＡＴＭ　ＡＲＰサーバ、２４…アドレス変換テーブル書き換え装置、２５…ホスト、２３－１…アドレス変換テーブル、２４－１…アドレス変換テーブル書き換えプログラム、２４－２…共通ＩＰアドレス／ＡＴＭアドレス対応テーブル、３１…ＩＰ通信網、３２…サーバ群、３２－１～３２－ｎ…サーバ、３３…ゲートウェイ・サーバ、３４…クライアント・ホスト、３３－１…対応テーブル、３３－２…接続先選択プログラム、３３－３…ＨＴＭＬファイル生成プログラム、３５…クライアント・ホストに返送されるＨＴＭＬファイル。

【図２】

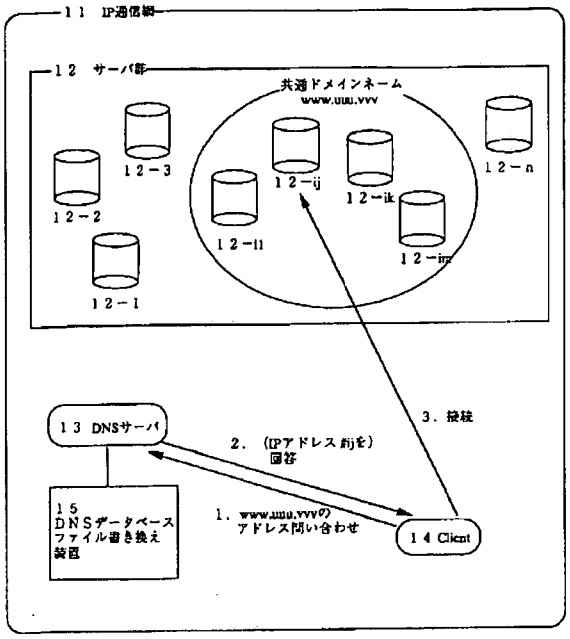


【図７】

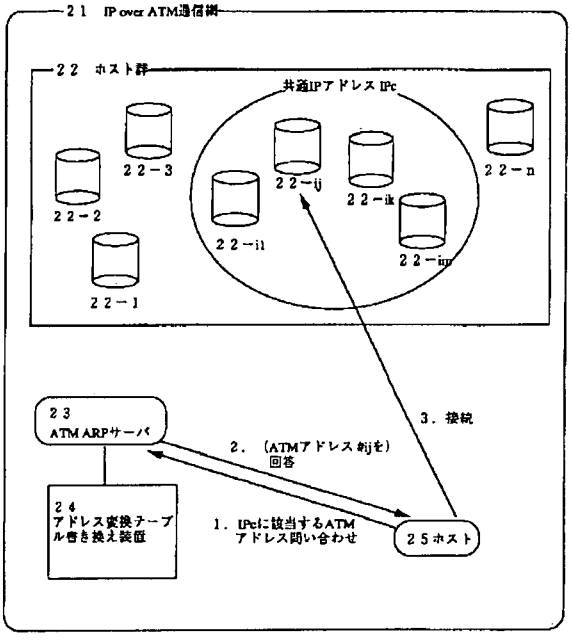


ゲートウェイ・サーバからクライアント・ホストに返送されるＨＴＭＬファイルの例

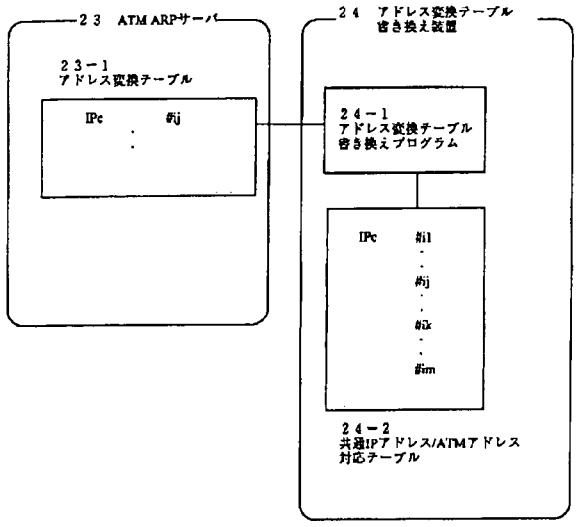
【図１】


１１　IP通信網
１２　サーバ群
共通ドメインネーム
www.uuu.vvv
１２－１
１２－２
１２－３
１２－ij
１２－ik
１２－il
１２－im
１２－n
１３ DNSサーバ
１４ Client
１５
DNSデータベース
ファイル書き換え
装置
１．www.uuu.vvvの
アドレス問い合わせ
２．（IPアドレス #ijを）
回答
３．接続

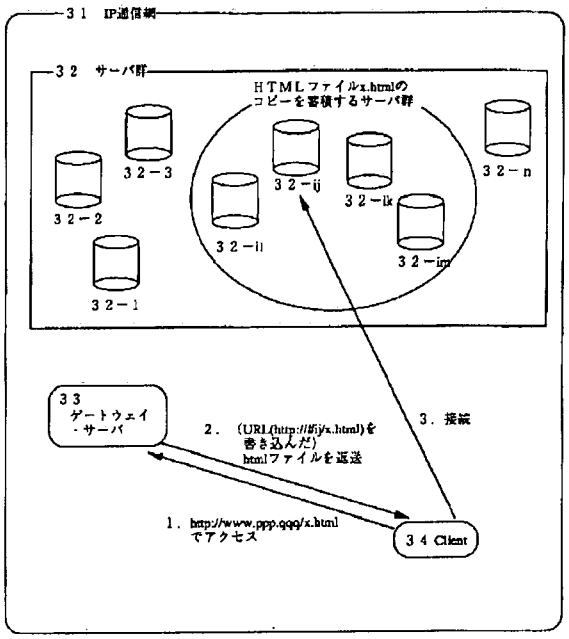
【図３】


２１　IP over ATM通信網
２２　ホスト群
共通IPアドレス IPc
２２－１
２２－２
２２－３
２２－ij
２２－ik
２２－il
２２－im
２２－n
２３
ATM ARPサーバ
２４
アドレス変換テーブ
ル書き換え装置
２５ホスト
１．IPcに該当するATM
アドレス問い合わせ
２．（ATMアドレス #ijを）
回答
３．接続

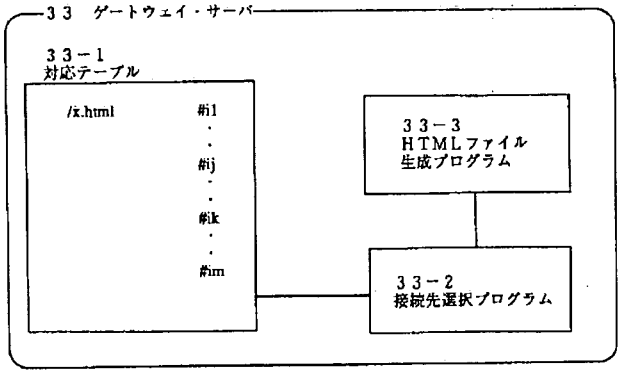
【図４】


２３　ATM ARPサーバ
２３－１
アドレス変換テーブル
IPc　#ij
２４　アドレス変換テーブル
書き換え装置
２４－１
アドレス変換テーブル
書き換えプログラム
IPc　#i1
#ij
#ik
#im
２４－２
共通IPアドレス/ATMアドレス
対応テーブル

【図５】



【図６】



フロントページの続き

(72)発明者　堀米　英明
東京都新宿区西新宿三丁目19番２号　日本
電信電話株式会社内

(72)発明者　五十川　裕
東京都新宿区西新宿三丁目19番２号　日本
電信電話株式会社内

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 10-093552
(43)Date of publication of application : 10.04.1998

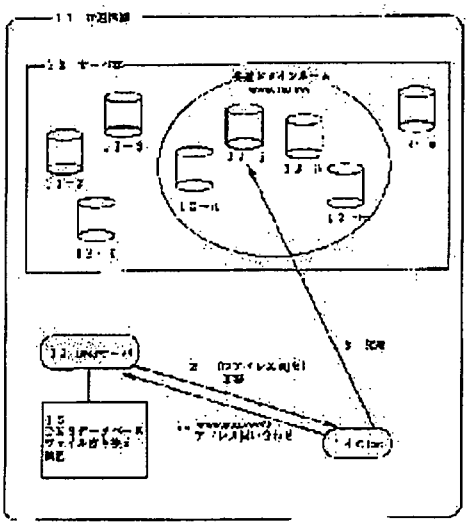
(51)Int.Cl. H04L 12/00
G06F 13/00

(21)Application number : 08-240114
(22)Date of filing : 11.09.1996

(71)Applicant : NIPPON TELEGR & TELEPH CORP <NTT>
(72)Inventor : MURAKAMI HIDEYO
HARA HIROYUKI
NAKAJIMA ISAMI
HORIGOME HIDEAKI
ISOGAWA YUTAKA

(54) COMMUNICATION CONNECTION METHOD WITH A PLURALITY OF HOSTS HAVING COMMON IDENTIFIER

(57)Abstract:
PROBLEM TO BE SOLVED: To maintain quality of continuous service by distributing the load on a faulty or overloaded host to other hosts so as to relive the load on the distributed source host and to change the communicating party without letting the client host to be aware of the change.
SOLUTION: A common identifier is provided to a server group 12 and in the case of communication of other client host 14 with the server group 12, the client uses the common identifier (common domain name) to access the server group 12, then a connection destination is changed to other host 12-ij in the server group 12 based on a connection destination revision algorithm 15 when an opposite party 12-i1 is faulty or overloaded so as to enable the transmission without making the client host 14 to be aware of the change.



LEGAL STATUS

[Date of request for examination] 15.12.2000
[Date of sending the examiner's decision of rejection] 06.01.2004
[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]
[Date of final disposal for application]
[Patent number]
[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

* NOTICES *

JPO and INPIT are not responsible for any
damages caused by the use of this translation.

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3.In the drawings, any words are not translated.

CLAIMS

[Claim(s)]
[Claim 1] In the communication link connection method of a communication network for two or more hosts connected through the communication network to communicate mutually It is installed in two or more locations, and a common identifier is given to the host group which consists of two or more physically different hosts. In this communication network, the conversion server changed into the identifier of each different host proper from this common identifier from the above-mentioned common identifier is installed. When sending from the host of the arbitration which is not contained in the above-mentioned host group to the host of the arbitration contained in this host group, The host of an origination side asks the above-mentioned conversion server using the above-mentioned common identifier. It is a communication link connection method with two or more hosts to whom this conversion server returns the host of an origination side the proper identifier of a destination-side host required for connection, and the host of this origination side has the common identifier characterized by making connection to the specific host in a host group based on the returned proper identifier.
[Claim 2] It is a communication link connection method with two or more hosts with the common identifier characterized by changing the specific host of a connection place by holding the correspondence table on which said conversion server matches a common identifier and the proper identifier of the specific host in the host group corresponding to this common identifier in a communication link connection method with two or more hosts with a common identifier according to claim 1, and rewriting according to the algorithm which was able to determine the this table beforehand.
[Claim 3] A communication link connection method with two or more hosts who have the common identifier characterized by changing a sequential connection place into the next host as the sequence beforehand assigned to the host when the number of accumulation connection reaches the specific number of connection set up beforehand as a rewriting algorithm of said correspondence table in a communication link connection method with two or more hosts with a common identifier according to claim 2.
[Claim 4] A communication link connection method with two or more hosts who have the common identifier characterized by changing a sequential connection place into the next host as a rewriting algorithm of said correspondence table in a communication link connection method with two or more hosts with a common identifier according to claim 2 as the sequence which was set up beforehand, and which was beforehand assigned to the host for every time interval.
[Claim 5] In a communication link connection method with two or more hosts with a common identifier according to claim 2 When the load of the specific host in the matched host group exceeds the threshold set up beforehand as a rewriting algorithm of said correspondence table, Or when failure occurs to this specific host, another host to whom it is operating normally as a host who replaces this host, and the operation load is not over the above-mentioned threshold is chosen. A communication link connection method with two or more hosts with the common identifier characterized by changing a connection place into the this selected host.

[Translation done.]

* NOTICES *

JPO and INPIT are not responsible for any
damages caused by the use of this translation.

1.This document has been translated by computer. So the translation may not reflect the original precisely.
2.**** shows the word which can not be translated.
3 In the drawings, any words are not translated

DETAILED DESCRIPTION

[Detailed Description of the Invention]
[0001]
[Field of the Invention] Even if this invention can distribute a load by distributing the connection request to the host to two or more of other duplicate hosts even when two or more hosts share a common identifier and an overload or failure arises to a host and a communications partner is changed, an origination-side host is related with a communication link connection method with two or more hosts with the common identifier which can be sent without being conscious of it.
[0002]
[Description of the Prior Art] Two or more hosts who interconnect through the communication network conventionally If an identifier (address) required in order to make connection using a destination-side host's identifier (name) currently exhibited is asked to a conversion server The conversion server had returned the origination-side host the identifier (address) required for a communication link with reference to the correspondence table which has described the correspondence relation of 1 to 1 between the identifier (name) currently exhibited and an identifier (address) required for a communication link The origination-side host is making connection with a destination-side host using an identifier (address) required for the returned communication link. However, when a majority of other hosts who make demands on a certain specific host for connection exist, the communication link to the host carries out congestion, and connection cannot be made
[0003]
[Problem(s) to be Solved by the Invention] Thus, in the conventional correspondence procedure between two or more hosts, when there are many hosts who expect connection of a certain specific host, the load of a channel linked to the specific host or its specific specific host became large and the connection request has exceeded those throughputs, the communication link to the host becomes difficult. In that case, possibility that the host will no longer be received also produces the communication link from other hosts who try access using the identifier which shows the host. Then, the purpose of this invention is to offer a communication link connection method with two or more hosts with the common identifier [it is possible to distribute the load of the host who solved such a conventional technical problem and became failure or an overload, and / omitting great plant-and-equipment investment] which can maintain a continuous and good quality of service. Moreover, other purposes of this invention are to offer a communication link connection method with two or more hosts with a common identifier with possible making it send, without making an origination-side host conscious of it, even if the identifier of a communications partner is changed.
[0004]
[Means for Solving the Problem] In case the host who gives a common identifier to the host group which consists of two or more hosts in a communication link connection method with two or more hosts who have the common identifier of this invention in order to attain the above-mentioned purpose, and is not contained in this host group communicates with the host contained in this host group, he makes connection using a proper identifier by asking the conversion server equipped with the function changed into a proper identifier from a common

identifier When a conversion server changed a connection place into other hosts in a host group, it enabled it to offer communication service continuously, without the specific host in a host group making conscious the host who is not contained in this host group, the case of failure or an overload, or even when a situation with a difficult communication link arises owing to failure of a communication network or congestion. Moreover, when working normally, it became possible to distribute a load by distributing connection with the host who is not contained in a host group within a host group. Therefore, the identifier of a proper which is different for every host required for a communication link in response to a common identifier within a communication network is asked to a common identifier and the identifier of a host proper based on (a) common identifier to the conversion server which grasps correspondence relation, the identifier of a proper is chosen as one host from two or more hosts who correspond to a common identifier by the conversion server, and the host who asked this is notified.
(b) When it detects that failure of the load and host who join a host by supervising the system operating status of the host to whom the same common identifier was given, and the communication link situation of the communication network linked to a host, or the traffic of the communication network linked to a host exceeded the specific threshold, a conversion server chooses a suitable host out of two or more hosts based on the above-mentioned information, and assigns the identifier of a proper.
[0005]
[Embodiment of the Invention] Hereafter, a drawing explains the example of this invention to a detail.
(The 1st example) Drawing 1 is the explanatory view of the communication link connection method between two or more hosts showing the 1st example of this invention, and drawing 2 is drawing showing the conversion table and rewriting equipment of the common identifier and the proper address in drawing 1 . drawing 1 -- as a common identifier -- a domain name -- moreover, the connection method at the time of using an IP address as a proper identifier, respectively, and using two or more servers is shown. Moreover, drawing 2 shows the functional configuration of a DNS (Domain Name System) server and DNS database file rewriting equipment. In this case, although which is applicable if it is Internet access service using a domain name, below, WWW (World Wide Web) is explained as an example. In drawing 1 , 11 is an IP communication network constituted by the IP router, and he is the client host whom the server group which 12 becomes from n sets of the WWW servers of 12-1, 12-2, ., 12-n, and 13 equipped with the Domain Name System server (DNS server), and 14 equipped with the WWW browser. A part of server group 12 surrounded with the circle supports IP address #i1 and .. from which server 12-i1 and .. which accumulate the copy of HTML file aaa.html, and im (1 <=m<=n) differ, respectively, and #im, and it shares domain name www.uuu.vvv.
[0006] In drawing 1 , the client host 14 performs access using URL (http://www.uuu.vvv/aaa.html) currently exhibited beforehand, in order to peruse aaa.html using a WWW browser. In that case, the client host 14 asks DNS server 13, in order for the IP address of the server applicable to domain name www.uuu.vvv to come to hand. In drawing 2 , when IP address #ij (1 <=j<=m) supports domain name www.uuu.vvv in the DNS database file 13-1 which DNS server 13 has at this time, DNS server 13 notifies the client host 14 of IP address #ij When it is received, the client host 14 uses IP address #ij, and is http to server 12-ij. Server 12-ij to delivery and aaa.html are downloaded for request. DNS database file rewriting equipment 15 consists of table files 15-2 corresponding to the common domain name / server IP address referred to by the DNS database file rewriting program 15-1 and its program 15-1, as shown in drawing 2 . Based on the connection place modification algorithm set up beforehand, the DNS database file rewriting program 15-1 operates, and the IP address corresponding to domain name www.uuu.vvv of the DNS database file 13-1 is changed into #ik (1 <=k<=m) from #ij. In this case, refer to the table 15-2 corresponding to a common domain name / IP address for the DNS database file rewriting program 15-1. All correspondences with community domain name www.uuu.vvv, and server 12-i1 and .. which use it, IP address #i1 of

12-im, ... and #im are memorized by the table 15-2 corresponding to a common domain name / IP address.
[0007] If after modification of the DNS database file 13-1 asks DNS server 13 the IP address applicable to domain name www.uuu.vvv in order that the client host 14 may download aaa.html, since IP address #ik will be notified as the reply, the client host 14 will access server 12-ik It is performed by the following procedures as an example of a connection place modification algorithm
(1) If the number of accumulation connection reaches the specific number of connection set up beforehand, the DNS database file 13-1 will be rewritten so that it may connect with 12-i1, 12-i2, ..., 12-im, 12-i1, and sequence like ....
(2) For every time interval set up beforehand, rewrite the DNS database file 13-1 so that it may connect with 12-i1, 12-i2, ..., 12-im, 12-i1, and sequence like ....
(3) When the load of server 12-ij exceeds the threshold set up beforehand, or when failure occurs in server 12-ij As a server which DNS database file rewriting equipment 15 replaces with server 12-ij in response to the notice of generating of those abnormalities from server 12-ij Server 12-ik to which it is operating normally and the operation load is not over the threshold is chosen, and the DNS database file 13-1 is rewritten to server 12-ik
[0008] (The 2nd example) Drawing 3 is the explanatory view of a communication link connection method with two or more hosts who show the 2nd example of this invention, and drawing 4 is the conversion table of the common identifier and the proper address in drawing 3 , and the block diagram of rewriting equipment. Here, the network at the time of applying this invention to the communication network which realizes IP routing on ATM (Asynchronous Transfer Mode) is shown And the case where a host with the ATM address which is different in an IP address common as a common identifier, respectively uses is explained. The IPover ATM communication network with which 21 is constituted by an ATM switch and the IP router in drawing 3 , the host group which 22 becomes from n sets of the hosts of 22-1, 22-2, . ., 22-n, and 23 are ATM. An ARP (Address ResolutionProtocol) server and 24 are ATM. The address translation table rewriting equipment which rewrites the address translation table of the ARP server 23, and 25 are hosts who are not contained in the host group 22. host 22-i1 and .... which the host group surrounded with the circle is a part of host group 22, and accumulate the copy of a certain file F -- ATM address #i1 from which 22-im (1 <=m<=n) differs, respectively .. it corresponds to #im and common IP address IPc is shared.
[0009] In drawing 3 , a host 25 tries access to the host who holds File F using ftp. It is ATM in order to obtain the ATM address of a communication link place required in order to open SVC (Switched Virtual Connection) among the hosts of a communication link place and to perform an ATM communication link to it in that case. The ATM address of the host applicable to IP address IPc is asked to the ARP server 23. At this time, it is ATM. It is ATM when ATM address #ij (1 <=j<=m) supports IP address IPc in the address translation table 23-1 which the ARP server 23 has. The ARP server 23 notifies a host 25 of ATM address #ij, and the host 25 who received it stretches SVC between host 22-ij(s) using ATM address #ij, and it downloads File F from host 22-ij. In drawing 4 , based on the connection place modification algorithm set up beforehand, the address translation table rewriting program 24-1 operates, and address translation table rewriting equipment 24 changes the ATM address corresponding to IP address IPc of an address translation table 23-1 into #ik (1 <=k<=m) from #ij. In this case, refer to a common IP address / the ATM address-mapping table 24-2 for the address translation table rewriting program 24-1. All correspondences with common IP address IPc, and host 22-i1 and .. which use it, ATM address #ij of 22-im, ... and #im are memorized by the common IP address / ATM address-mapping table 24-2. Since ATM address #ik will be notified as a reply if the host's ATM address is asked to the ATMARP server 23 in order that after modification of an address translation table 23-1 may communicate with the host to whom a host 25 corresponds to IP address IPc, a host 25 will perform a communication link with host 22-ik. In addition, as a connection place modification algorithm, it is the same as (1) - (3) stated in the 1st example.
[0010] (The 3rd example) Drawing 5 is the explanatory view of a communication link connection

method with two or more hosts who show the 3rd example of this invention, drawing 6 is the conversion table of the common identifier and the proper address in drawing 5 , and the block diagram of rewriting equipment, and drawing 7 is drawing showing the example of the HTML file returned to a client host from the gateway server in drawing 5 . The case where the gateway server (GW) representing the server group which shows the connection method in the case of here applying this invention to offer of the WWW (World Wide Web) service in IP communication network which consists of two or more routers which used Internet Protocol, and consists a domain name common as a common identifier of two or more WWW servers uses is shown. In drawing 5 , IP communication network with which 31 is constituted by the IP router, the server group which 32 becomes from n sets of the WWW servers of 32-1, 32-2, ..., 32-n, the gateway server (GW) to which 33 receives the connection request to the server group 32, and 34 are the client hosts having a WWW browser. The server group surrounded with the circle corresponds to IP address #i1 and .. from which server 32-i1 and ... which are a part of server group 32, and accumulate the copy of HTML file x.html, and 32-im (1 <=m<=n) differ, respectively, and #im.

[0011] The client host 34 accesses the gateway server 33 using URL (http://www.ppp.qqq/x.html) using domain name www.ppp.qqq of the gateway server 33 currently exhibited beforehand first, in order to peruse x.html using a WWW browser. The gateway server 33 consists of a correspondence table 33-1, a HTML file-generating program 33-3, and a connection place selection program 33-2, as shown in drawing 6 . Correspondence with HTML file x.html, and IP address #i1 of the server which accumulates it, ... and #im is described by the correspondence table 33-1. The connection place selection program 33-2 chooses IP address #ij of server 32-ij (1 <=j<=m) based on the connection place selection algorithm beforehand set up with reference to the correspondence table 33-1. The HTML file-generating program 33-3 creates HTML file 35 as shown in drawing 7 based on this selection, and returns the client host 34 it. The client host 34 will perform access to server 32-ik for the html file in which URL (http://#ij/x.html) using IP address #ij currently written in it was written using reception and it, if HTML file 35 is received. In addition, as a connection place modification algorithm, it is the same as the example stated in the 1st above-mentioned example.

[0012]

[Effect of the Invention] As explained above, when failure or an overload arises to (a) distribution former host according to this invention Since a distributing agency host's load can be distributed by arranging a distributing agency host's duplicate to other hosts, and distributing the connection request to a distributing agency host to two or more duplicate hosts A continuous and good quality of service can be maintained without a distributing agency host owner doing great plant-and-equipment investment beforehand for an overload with difficult prediction, or failure. moreover, (b) -- it can perform by sharing a common identifier between two or more hosts, without making an origination-side host conscious of modification of a communications partner.

[Translation done.]

(19)日本国特許庁（ＪＰ）　(12) 公開特許公報（A）　(11)特許出願公開番号

特開平10－93552

(43)公開日　平成10年(1998)４月10日

(51)Int.Cl.[6]	識別記号	ＦＩ	
Ｈ０４Ｌ　12/00		Ｈ０４Ｌ　11/00	
Ｇ０６Ｆ　13/00	３５５	Ｇ０６Ｆ　13/00	３５５

審査請求　未請求　請求項の数５　ＯＬ　（全７頁）

(21)出願番号　特願平8－240114

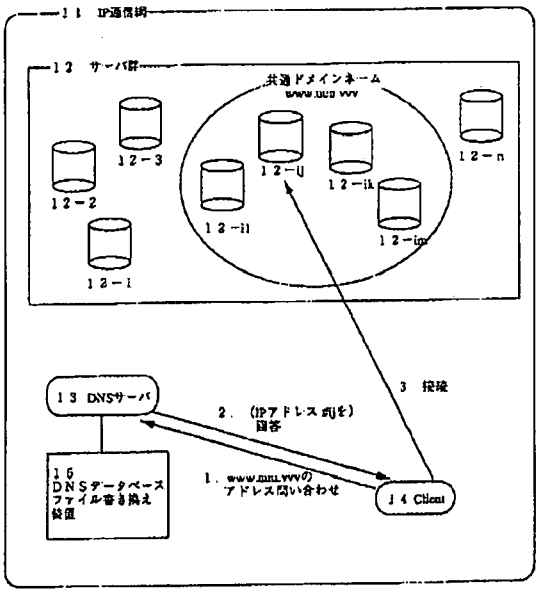
(22)出願日　平成８年(1996)９月11日

(71)出願人　000004226
日本電信電話株式会社
東京都新宿区西新宿三丁目19番２号
(72)発明者　村上　英世
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内
(72)発明者　原　博之
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内
(72)発明者　中島　伊佐美
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内
(74)代理人　弁理士　磯村　雅俊　（外１名）

最終頁に続く

(54)【発明の名称】　共通識別子を持つ複数ホストとの通信接続方法

(57)【要約】
【課題】故障または過負荷のホストを他のホストに分散して、分散元ホストの負荷を軽減することができ、継続的サービス品質を維持する。また、発信側ホストに意識させずに、通信相手を変更できるようにする。
【解決手段】共通識別子をサーバ群１２に付与し、それ以外のホスト１４がサーバ群１２と通信する場合には、共通識別子（共通ドメインネーム）を用いてアクセスすると、相手（１２－ｉ１）が故障または過負荷の場合には、接続先変更アルゴリズム１５に基づいてサーバ群１２内の他のホスト（１２－ｉｊ）に接続先を変更することにより、変更を意識させずに発信できるようにする。



【特許請求の範囲】
【請求項1】通信網を介して接続された複数のホストが、相互間で通信するための通信網の通信接続方法において、
複数の場所に設置され、物理的に異なる複数のホストからなるホスト群に対して共通の識別子を付与し、
該通信網内に、上記共通識別子から該共通識別子とは異なる各ホスト固有の識別子に変換する変換サーバを設置し、
上記ホスト群に含まれない任意のホストから、該ホスト群に含まれる任意のホストに対して発信する場合、発信側のホストは上記共通識別子を用いて上記変換サーバに問い合わせを行い、
該変換サーバは接続に必要な着信側ホストの固有識別子を発信側のホストに返送し、
該発信側のホストは返送された固有識別子を基にホスト群中の特定のホストに接続を実行することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【請求項2】請求項1に記載の共通識別子を持つ複数ホストとの通信接続方法において、
前記変換サーバは、共通識別子と、該共通識別子に対応するホスト群中の特定のホストの固有識別子とを対応づける対応テーブルを保持し、
該対応テーブルを予め決められたアルゴリズムに従って書き換えることにより、接続先の特定のホストを変更することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【請求項3】請求項2に記載の共通識別子を持つ複数ホストとの通信接続方法において、
前記対応テーブルの書き換えアルゴリズムとして、累積接続数が予め設定しておいた特定の接続数に達すると、予めホストに割り振られた順番通りに次のホストに順次接続先を変更することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【請求項4】請求項2に記載の共通識別子を持つ複数ホストとの通信接続方法において、
前記対応テーブルの書き換えアルゴリズムとして、予め設定しておいた時間間隔毎に、予めホストに割り振られた順番通りに次のホストに順次接続先を変更することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【請求項5】請求項2に記載の共通識別子を持つ複数ホストとの通信接続方法において、
前記対応テーブルの書き換えアルゴリズムとして、対応付けられたホスト群中の特定のホストの負荷が予め設定しておいた閾値を超えた場合、あるいは該特定のホストに故障が発生した場合に、該ホストに代るホストとして正常に作動しており、かつ稼働負荷が上記閾値を超えていない別のホストを選択して、該選択されたホストに接続先を変更することを特徴とする共通識別子を持つ複数ホストとの通信接続方法。
【発明の詳細な説明】
【0001】
【発明の属する技術分野】本発明は、共通の識別子を複数のホストが共有することにより、ホストに過負荷または故障が生じたときでも、そのホストに対する接続要求を複数の他の複製ホストに分配することにより、負荷を分散させることが可能であり、かつ通信相手が変更されても発信側ホストはそれを意識せずに発信することが可能な共通識別子を持つ複数ホストとの通信接続方法に関する。
【0002】
【従来の技術】従来、通信網を介して相互接続されている複数のホストは、公開されている着信側ホストの識別子（ネーム）を用いて接続を実行するために必要な識別子（アドレス）を変換サーバに問い合わせると、変換サーバは公開されている識別子（ネーム）と通信に必要な識別子（アドレス）との1対1の対応関係を記述している対応テーブルを参照して、通信に必要な識別子（アドレス）を発信側ホストに返送していた。発信側ホストは、返送された通信に必要な識別子（アドレス）を用いて、着信側ホストとの接続を実行している。しかしながら、ある特定のホストに接続を要求する他のホストが多数存在する場合には、そのホストへの通信が輻輳してしまい、接続が実行できない。
【0003】
【発明が解決しようとする課題】このように、従来の複数ホスト相互間の通信方法では、ある特定のホストに接続を希望するホストが多い場合、その特定のホストあるいはその特定のホストに接続する通信路の負荷が大きくなり、接続要求がそれらの処理能力を超えてしまった時点で、そのホストへの通信が困難となる。その場合には、そのホストを示す識別子を用いてアクセスを試みる他のホストからの通信を、そのホストが受け付けられなくなる可能性も生じてくる。そこで、本発明の目的は、このような従来の課題を解決し、故障あるいは過負荷となったホストの負荷を分散させることが可能であり、多大な設備投資を省略しながら、継続的かつ良好なサービス品質を維持することが可能な共通識別子を持つ複数ホストとの通信接続方法を提供することにある。また、本発明の他の目的は、通信相手の識別子が変更になっても、発信側ホストにそれを意識させることなく発信させることが可能な共通識別子を持つ複数ホストとの通信接続方法を提供することにある。
【0004】
【課題を解決するための手段】上記目的を達成するため、本発明の共通識別子を持つ複数ホストとの通信接続方法では、共通識別子を複数のホストからなるホスト群に付与し、該ホスト群に含まれないホストは該ホスト群に含まれるホストと通信する際には、共通識別子から固

有識別子に変換する機能を備えた変換サーバに問い合わせることにより、固有識別子を用いて接続を実行する。ホスト群中の特定のホストが故障や過負荷の場合、あるいは通信網の故障や輻輳が原因で通信が困難な状況が生じた場合でも、該ホスト群に含まれないホストに意識させることなく、変換サーバがホスト群内の他のホストに接続先を変更することにより、継続的に通信サービスを提供することができるようにした。また、正常に稼働している場合においても、ホスト群内でホスト群に含まれないホストとの接続を分散させることにより、負荷を分散させることが可能になった。そのために、（ａ）共通識別子を基に、通信網内で共通識別子を受けて通信に必要な各ホスト毎に異なる固有の識別子を、共通識別子とホスト固有の識別子に対応関係を把握している変換サーバに対して問い合わせ、変換サーバで共通識別子に該当する複数のホストの中から１つのホストに固有の識別子を選択し、これを問い合わせたホストに通知する。

（ｂ）変換サーバは、同一の共通識別子が付与されたホストの稼働状況、ホストに接続する通信網の通信状況を監視することにより、ホストに加わる負荷、ホストの故障、あるいはホストに接続する通信網の通信量が特定の閾値を超えたことを検出したとき、上記情報に基づいて複数のホストの中から適切なホストを選択し、固有の識別子を割り当てる。

【０００５】

【発明の実施の形態】以下、本発明の実施例を、図面により詳細に説明する。

（第１実施例）図１は、本発明の第１の実施例を示す複数ホスト相互間の通信接続方法の説明図であり、図２は図１における共通識別子と固有アドレスとの対応表および書き換え装置を示す図である。図１では、共通識別子としてドメインネームを、また固有識別子としてＩＰアドレスをそれぞれ用い、かつ複数のサーバを用いた場合の接続方法を示している。また図２では、ＤＮＳ（ドメインネームシステム）サーバおよびＤＮＳデータベースファイル書き換え装置の機能構成を示している。この場合、ドメイン・ネームを用いるインターネットアクセスサービスであれば、どれでも適用することができるが、以下ではWWW（World Wide Web)を例として説明する。図１において、１１はＩＰルータにより構成されるＩＰ通信網であり、１２は１２－１，１２－２，・・，１２－ｎのｎ台のWWWサーバからなるサーバ群、１３はドメイン・ネーム システム・サーバ（ＤＮＳサーバ）、１４はWWWブラウザを備えたクライアント・ホストである。円で囲まれているサーバ群１２の一部は、ＨＴＭＬファイルａａａ．ｈｔｍｌのコピーを蓄積するサーバ１２－ｉ１，・・・，ｉｍ（１≦ｍ≦ｎ）がそれぞれ異なるＩＰアドレス＃ｉ１，・・・，＃ｉｍに対応しており、かつドメイン ネームwww．ｕｕｕ．ｖｖｖを共有する。

【０００６】図１において、クライアント・ホスト１４は、WWWブラウザを用いてａａａ．ｈｔｍｌを閲覧するために、予め公開されているＵＲＬ（ｈｔｔｐ：／／ｗｗｗ．ｕｕｕ．ｖｖｖ／ａａａ．ｈｔｍｌ）を用いてアクセスを実行する。その際に、クライアント・ホスト１４は、ドメイン ネームwww．ｕｕｕ．ｖｖｖに該当するサーバのＩＰアドレスを入手するために、ＤＮＳサーバ１３に問い合わせを行う。図２において、この時、ＤＮＳサーバ１３が持つＤＮＳデータベースファイル１３－１において、ドメイン・ネームwww．ｕｕｕ．ｖｖｖにＩＰアドレス＃ｉｊ（１≦ｊ≦ｍ）が対応している場合、ＤＮＳサーバ１３はクライアント・ホスト１４にＩＰアドレス＃ｉｊを通知する。クライアント・ホスト１４は、それを受け取ると、ＩＰアドレス＃ｉｊを用いてサーバ１２－ｉｊにｈｔｔｐ ｒｅｑｕｅｓｔを送り、サーバ１２－ｉｊからａａａ．ｈｔｍｌをダウンロードする。ＤＮＳデータベースファイル書き換え装置１５は、図２に示すように、ＤＮＳデータベースファイル書き換えプログラム１５－１と、そのプログラム１５－１により参照される共通ドメインネーム／サーバＩＰアドレス対応テーブルファイル１５－２から構成される。予め設定された接続先変更アルゴリズムに基づいてＤＮＳデータベースファイル書き換えプログラム１５－１が作動し、ＤＮＳデータベースファイル１３－１のドメイン・ネームwww．ｕｕｕ ｖｖｖに対応するＩＰアドレスを＃ｉｊから＃ｉｋ（１≦ｋ≦ｍ）に変更する。この際に、ＤＮＳデータベースファイル書き換えプログラム１５－１は、共通ドメイン・ネーム／ＩＰアドレス対応テーブル１５－２を参照する。共通ドメイン・ネーム／ＩＰアドレス対応テーブル１５－２には、共通ドメイン ネームwww ｕｕｕ．ｖｖｖと、それを用いているサーバ１２－ｉ１，・・・，１２－ｉｍのＩＰアドレス＃ｉ１，・・・，＃ｉｍとの対応が全て記憶されている。

【０００７】ＤＮＳデータベースファイル１３－１の変更後は、クライアント・ホスト１４がａａａ．ｈｔｍｌをダウンロードするために、ドメイン・ネームwww．ｕｕｕ ｖｖｖに該当するＩＰアドレスをＤＮＳサーバ１３に問い合わせると、その回答としてＩＰアドレス＃ｉｋが通知されるので、クライアント・ホスト１４はサーバ１２－ｉｋにアクセスすることになる。接続先変更アルゴリズムの例としては、次のような手順で行われる。

（１）累積接続数が予め設定しておいた特定の接続数に達すると、１２－ｉ１，１２－ｉ２，・・・，１２－ｉｍ，１２－ｉ１，・・・のような順番に接続するように、ＤＮＳデータベースファイル１３－１を書き換える。

（２）予め設定しておいた時間間隔毎に、１２－ｉ１，１２－ｉ２，・・，１２－ｉｍ，１２－ｉ１，・・

のような順番に接続するように、ＤＮＳデータベースファイル１３－１を書き換える。

（３）サーバ１２－ｉｊの負荷が予め設定しておいた閾値を超えた場合、あるいはサーバ１２－ｉｊに故障が発生した場合には、サーバ１２－ｉｊからそれらの異常の発生の通知をＤＮＳデータベースファイル書き換え装置１５が受けて、サーバ１２－ｉｊに代わるサーバとして、正常に作動しており、かつ稼働負荷が閾値を超えていないサーバ１２－ｉｋを選択し、ＤＮＳデータベースファイル１３－１をサーバ１２－ｉｋに書き換える。

【０００８】（第２実施例）図３は、本発明の第２の実施例を示す複数ホストとの通信接続方法の説明図であり、図４は、図３における共通識別子と固有アドレスとの対応表および書き換え装置の構成図である。ここでは、本発明をＡＴＭ（Asynchronous Transfer Mode)上でＩＰルーチングを実現する通信網に適用した場合のネットワークを示している。そして、共通識別子として共通のＩＰアドレスを、それぞれ異なるＡＴＭアドレスを持つホストが用いる場合を説明する。図３において、２１はＡＴＭスイッチおよびＩＰルータにより構成されるＩＰｏｖｅｒ ＡＴＭ通信網、２２は２２－１，２２－２，・・・，２２－ｎのｎ台のホストからなるホスト群、２３はＡＴＭ ＡＲＰ（Address Resolution Protocol)サーバ、２４はＡＴＭ ＡＲＰサーバ２３のアドレス変換テーブルを書き換えるアドレス変換テーブル書き換え装置、２５はホスト群２２に含まれないホストである。円で囲まれているホスト群は、ホスト群２２の一部であって、あるファイルＦのコピーを蓄積するホスト２２－ｉ１，・・・２２－ｉｍ（１≦ｍ≦ｎ）がそれぞれ異なるＡＴＭアドレス＃ｉ１・・・＃ｉｍに対応し、かつ共通ＩＰアドレスＩＰｃを共有している。

【０００９】図３において、ホスト２５は、ｆｔｐを用いてファイルＦを保持するホストにアクセスを試みる。その際に、通信先のホストとの間にＳＶＣ（Switched Virtual Connection)を開いてＡＴＭ通信を行うために必要な通信先のＡＴＭアドレスを得るために、ＡＴＭ ＡＲＰサーバ２３にＩＰアドレスＩＰｃに該当するホストのＡＴＭアドレスの問い合わせを行う。このとき、ＡＴＭ ＡＲＰサーバ２３が持つアドレス変換テーブル２３－１において、ＩＰアドレスＩＰｃにＡＴＭアドレス＃ｉｊ（１≦ｊ≦ｍ）が対応している場合、ＡＴＭ ＡＲＰサーバ２３はホスト２５にＡＴＭアドレス＃ｉｊを通知し、それを受け取ったホスト２５はＡＴＭアドレス＃ｉｊを用いてホスト２２－ｉｊとの間にＳＶＣを張り、ファイルＦをホスト２２－ｉｊよりダウンロードする。図４において、アドレス変換テーブル書き換え装置２４は、予め設定された接続先変更アルゴリズムに基づいてアドレス変換テーブル書き換えプログラム２４－１が作動し、アドレス変換テーブル２３－１のＩＰアドレスＩＰｃに対応するＡＴＭアドレスを＃ｉｊから＃ｉｋ（１≦ｋ≦ｍ）に変更する。この際に、アドレス変換テーブル書き換えプログラム２４－１は、共通ＩＰアドレス／ＡＴＭアドレス対応テーブル２４－２を参照する。共通ＩＰアドレス／ＡＴＭアドレス対応テーブル２４－２には、共通ＩＰアドレスＩＰｃと、それを用いているホスト２２－ｉ１，・・・，２２－ｉｍのＡＴＭアドレス＃ｉｊ，・・・，＃ｉｍとの対応が全て記憶されている。アドレス変換テーブル２３－１の変更後は、ホスト２５がＩＰアドレスＩＰｃに該当するホストと通信を行うために、そのホストのＡＴＭアドレスをＡＴＭ ＡＲＰサーバ２３に問い合わせると、回答としてＡＴＭアドレス＃ｉｋが通知されるので、ホスト２５はホスト２２－ｉｋと通信を行うことになる。なお、接続先変更アルゴリズムとしては、第１実施例で述べた（１）～（３）と同じである。

【００１０】（第３実施例）図５は、本発明の第３の実施例を示す複数ホストとの通信接続方法の説明図であり、図６は、図５における共通識別子と固有アドレスとの対応表および書き換え装置の構成図であり、図７は、図５におけるゲートウェイサーバからクライアント・ホストに返送されるＨＴＭＬファイルの例を示す図である。ここでは、本発明をインターネットプロトコルを用いた複数のルータで構成されるＩＰ通信網におけるＷＷＷ（World Wide Web)サービスの提供に適用する場合の接続方法を示しており、かつ共通識別子として共通のドメイン・ネームを、複数のＷＷＷサーバからなるサーバ群を代表するゲートウェイ・サーバ（ＧＷ）が用いる場合を示している。図５において、３１はＩＰルータにより構成されるＩＰ通信網、３２は３２－１，３２－２，・・・，３２－ｎのｎ台のＷＷＷサーバからなるサーバ群、３３はサーバ群３２への接続要求を受け付けるゲートウェイ・サーバ（ＧＷ）、３４はＷＷＷブラウザを備えたクライアント・ホストである。円で囲まれているサーバ群は、サーバ群３２の一部であり、かつＨＴＭＬファイルｘ．ｈｔｍｌのコピーを蓄積するサーバ３２－ｉ１，・・・３２－ｉｍ（１≦ｍ≦ｎ）がそれぞれ異なるＩＰアドレス＃ｉ１，・・・，＃ｉｍに対応する。

【００１１】クライアント・ホスト３４は、ＷＷＷブラウザを用いてｘ ｈｔｍｌを閲覧するために、初めに予め公開されているゲートウェイ・サーバ３３のドメイン・ネームｗｗｗ ｐｐｐ．ｑｑｑを用いたＵＲＬ（ｈｔｔｐ：／／ｗｗｗ．ｐｐｐ．ｑｑｑ／ｘ．ｈｔｍｌ）を用いてゲートウェイ・サーバ３３にアクセスする。ゲートウェイ サーバ３３は、図６に示すように、対応テーブル３３－１とＨＴＭＬファイル生成プログラム３３－３と接続先選択プログラム３３－２とから構成されている。対応テーブル３３－１には、ＨＴＭＬファイルｘ．ｈｔｍｌとそれを蓄積するサーバのＩＰアドレス＃ｉ１，　　，＃ｉｍとの対応が記述されている。接続先

選択プログラム３３－２は、対応テーブル３３－１を参照して予め設定されている接続先選択アルゴリズムに基づいてサーバ３２－ｉｊ（１≦ｊ≦ｍ）のＩＰアドレス＃ｉｊを選択する。ＨＴＭＬファイル生成プログラム３３－３は、この選択を基に図７に示すようなＨＴＭＬファイル３５を作成し、それをクライアント・ホスト３４に返送する。クライアント・ホスト３４は、ＨＴＭＬファイル３５を受け取ると、それに書き込まれているＩＰアドレス＃ｉｊを用いたＵＲＬ（ｈｔｔｐ：／／＃ｉｊ／ｘ　ｈｔｍｌ）が書き込まれたｈｔｍｌファイルを受け取り、それを用いてサーバ３２－ｉｋへのアクセスを実行する。なお、接続先変更アルゴリズムとしては、前述の第１実施例で述べた例と同じである。

【００１２】

【発明の効果】以上説明したように、本発明によれば、（ａ）分散元ホストに故障あるいは過負荷が生じたときには、他のホストに分散元ホストの複製を配置して、分散元ホストに対する接続要求を複数の複製ホストに分配することにより、分散元ホストの負荷を分散させることができるので、予め予測困難な過負荷や故障のために分散元ホスト所有者が多大な設備投資をすることなく、継続的かつ良好なサービス品質を維持することができる。また（ｂ）共通の識別子を複数のホストで共有することにより、通信相手の変更を発信側ホストに意識させることなく実行することができる。

【図面の簡単な説明】

【図１】本発明の第１の実施例を示す複数ホストとの通信接続方法の説明図である。

【図２】図１における共通識別子と固有アドレスとの対応表および書き換え装置の構成図である。

【図３】本発明の第２の実施例を示す複数ホストとの通信接続方法の説明図である。

【図４】図３における共通識別子と固有アドレスとの対応表および書き換え装置の構成図である。

【図５】本発明の第３の実施例を示す複数ホストとの通信接続方法の説明図である。

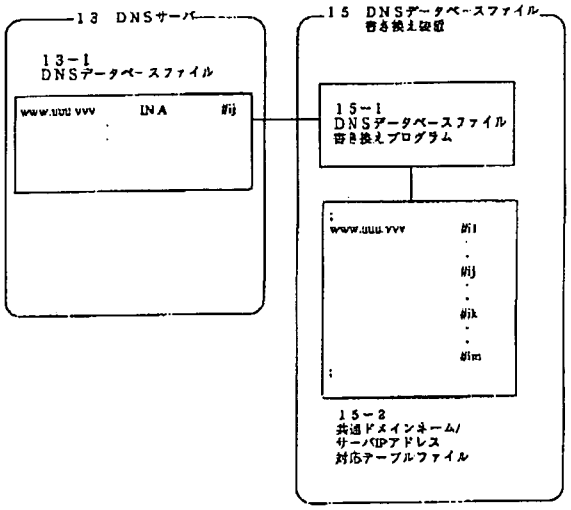
【図６】図５における共通識別子と固有アドレスとの対応表および書き換え装置の構成図である。

【図７】図５におけるゲートウェイサーバからクライアント・ホストに返送されるＨＴＭＬファイルの一例を示す図である。

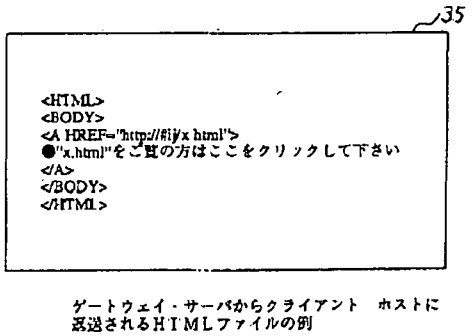
【符号の説明】

１１…ＩＰ通信網、１２…サーバ群、１２－１～１２－Ｎ…サーバ、１３…ＤＮＳサーバ、１４…クライアント・ホスト、１５…ＤＮＳデータベースファイル書き換え装置、１３－１…ＤＮＳデータベースファイル、１５－１…ＤＮＳデータベースファイル書き換えプログラム、１５－２…共通ドメインネーム／サーバＩＰアドレス対応テーブルファイル、２１…ＩＰｏｖｅｒＡＴＭ通信網、２２…ホスト群、２２－１～２２－ｎ…ホスト、２３…ＡＴＭ　ＡＲＰサーバ、２４…アドレス変換テーブル書き換え装置、２５…ホスト、２３－１…アドレス変換テーブル、２４－１…アドレス変換テーブル書き換えプログラム、２４－２…共通ＩＰアドレス／ＡＴＭアドレス対応テーブル、３１…ＩＰ通信網、３２…サーバ群、３２－１～３２－ｎ…サーバ、３３…ゲートウェイ・サーバ、３４…クライアント・ホスト、３３－１…対応テーブル、３３－２…接続先選択プログラム、３３－３…ＨＴＭＬファイル生成プログラム、３５…クライアント・ホストに返送されるＨＴＭＬファイル。

【図２】

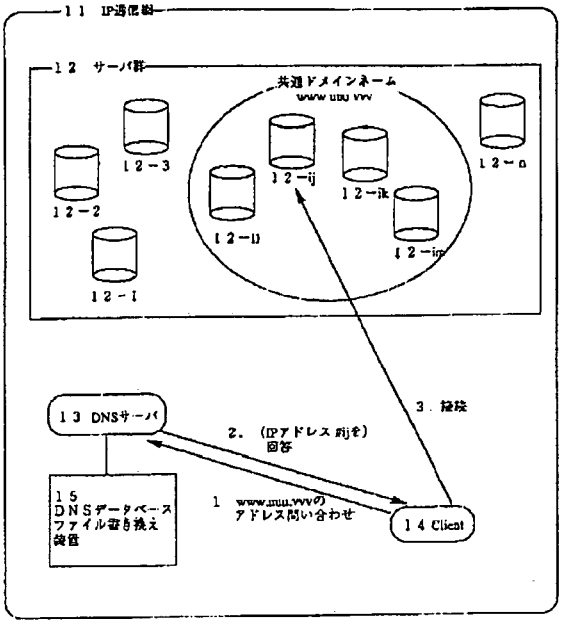


【図７】

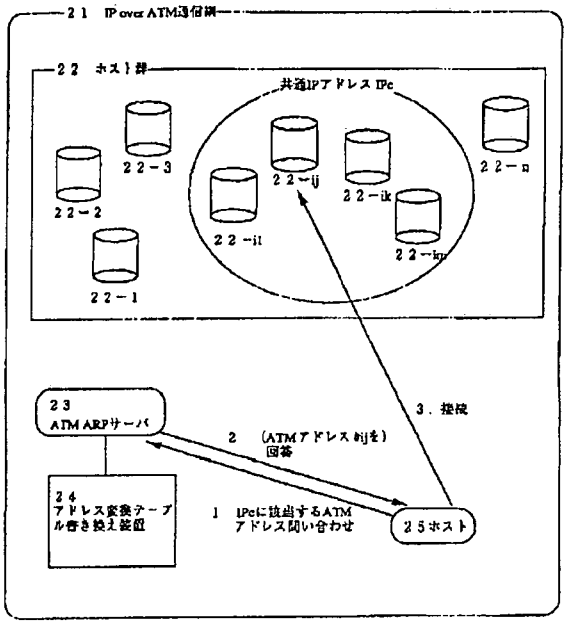


ゲートウェイ・サーバからクライアント　ホストに返送されるＨＴＭＬファイルの例

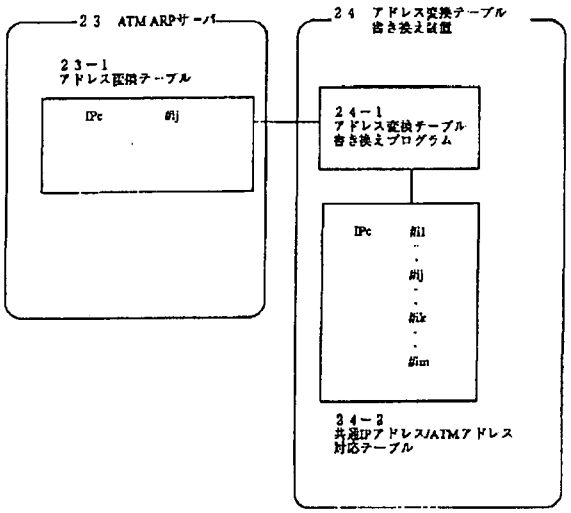
【図1】


11 IP通信網
12 サーバ群
共通ドメインネーム
www.uuu.vvv
12－1
12－2
12－3
12－ij
12－ik
12－n
13 DNSサーバ
2. (IPアドレス #ijを)
回答
3. 接続
15
DNSデータベース
ファイル書き換え
装置
1 www.uuu.vvvの
アドレス問い合わせ
14 Client

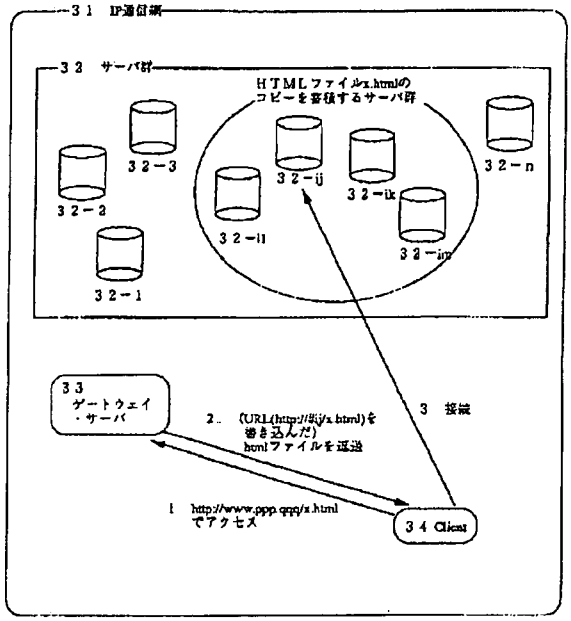
【図3】


21 IP over ATM通信網
22 ホスト群
共通IPアドレス IPc
22－1
22－2
22－3
22－il
22－ij
22－ik
22－n
23
ATM ARPサーバ
2 (ATMアドレス #ijを)
回答
3. 接続
24
アドレス変換テーブル
書き換え装置
1 IPcに該当するATM
アドレス問い合わせ
25ホスト

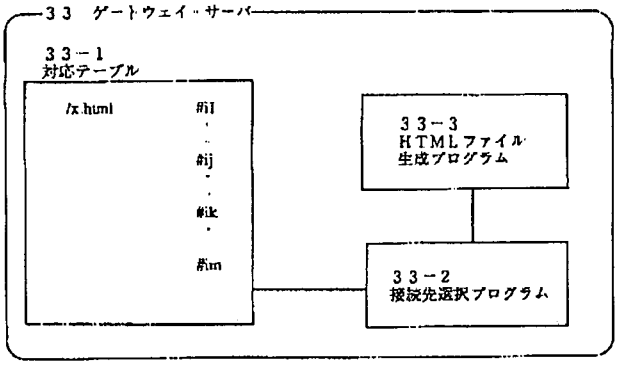
【図4】


23 ATM ARPサーバ
23－1
アドレス変換テーブル
IPc
#ij
24 アドレス変換テーブル
書き換え装置
24－1
アドレス変換テーブル
書き換えプログラム
IPc
#il
#ij
#ik
#im
24－2
共通IPアドレス/ATMアドレス
対応テーブル

【図5】



【図6】



フロントページの続き

(72)発明者　堀米　英明
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内

(72)発明者　五十川　裕
東京都新宿区西新宿三丁目19番２号　日本電信電話株式会社内

(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 1 063 831 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
27.12.2000 Bulletin 2000/52

(51) Int Cl.[7]: **H04L 29/06**

(21) Application number: **00305308.9**

(22) Date of filing: **23.06.2000**

(84) Designated Contracting States:
AT BE CH CY DE DK ES FI FR GB GR IE IT LI LU MC NL PT SE
Designated Extension States:
AL LT LV MK RO SI

(30) Priority: **24.06.1999 JP 17847499**

(71) Applicant: **CANON KABUSHIKI KAISHA**
Tokyo (JP)

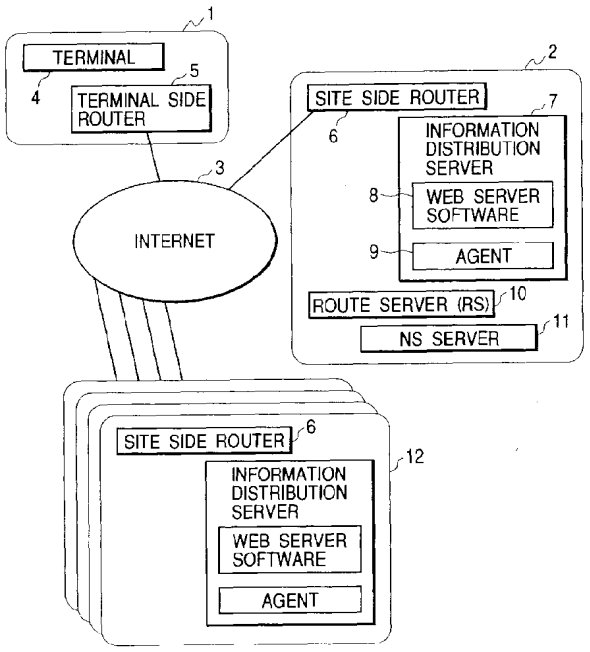
(72) Inventor: **Ogino, Tsukasa,**
Canon Kabushiki Kaisha
Tokyo (JP)

(74) Representative:
Beresford, Keith Denis Lewis et al
BERESFORD & Co.
High Holborn
2-5 Warwick Court
London WC1R 5DJ (GB)

(54) Network status server, information distribution system, control method, and storage medium for storing control program

(57) There is disclosed a network status server for receiving an inquiry from a web server accessed by a client, and using path information and determining the optimum web server to return a response to the accessed web server. The web server receiving the response instructs the client to access the optimum web server from the next time. In this constitution, accesses among the dispersed/arranged web server sites are leveled, and access loads can be leveled even among the web servers in the respective web server sites. Moreover, the client can access the optimum web server.

FIG. 1



EP 1 063 831 A2

Printed by Jouve, 75001 PARIS (FR)

Description

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] The present invention relates to a network status server for distributing large-capacity data on a network, an information distribution system, a control method, and a storage medium for storing a control program.

[0002] Particularly, the present invention relates to a network status server, an information distribution system, a control method and a storage medium for storing a control program when a plurality of dispersed/arranged information distribution servers are controlled.

Related Background Art

[0003] As one of information distribution techniques using Internet, World Wide Web (hereinafter referred to as Web) has generally been used. This technique comprises: operating a web server software on a general computer (information processor); and distributing information such as contents (characters, charts, tables, and the like) described in an HTML language. The computer as a client which accesses a server (hereinafter referred to as the access client) may be connected to the server via a network, and free perusal is possible with a general-purpose perusal software (browser). In recent years, a web server site accessible to the entire world has appeared and the importance has further increased. On the other hand, when the web server has been accessed from the entire world in a concentrated manner, a large load is applied to the connection network, and the web server. To solve the problem, several techniques are currently proposed as countermeasures. The techniques will briefly be described with reference to Figs. 6 and 7.

[0004] First, a system in which a virtual host computer is utilized as shown in Fig. 6 is proposed as a technique of dispersing accesses concentrated to the web server. This aims at dispersing the above-described load to a plurality of servers. In this example, it is supposed that the access client accesses URL (access address) of www.abcde.jp. In the access to www.abcde.jp, first the virtual host computer is accessed. The virtual host computer intends to appropriately disperse the accesses from the load situation of the respective subordinate web servers. Actually, any one of www1, www2, www3 distributes predetermined information in response to the access from the client, but the virtual host computer once accepts the access to appropriately disperse the access. Moreover, a transfer system by TCP connection hop is proposed. The system will be described with reference to Fig. 7. When there is an access from the client, first www1 is accessed. In the www1, a scheduler operates, and selects the optimum web server www3, and responds to the access from the client. Specifically, the www1 constantly responds to an access request, and the optimum web server (www1, www2, www3) responds. In this system, the access dispersion is intended in the schedule function.

[0005] However, the above-described conventional access dispersion system has the following problems.

1) When the virtual host computer shown in Fig. 6 is utilized, a problem occurs that a device serving as an exclusive virtual host computer is separately necessary.

2) Moreover, when the exclusive virtual host computer temporarily accepts all the accesses, and when there are many accesses, the processing ability of the virtual host computer raises a problem. Specifically, the processing ability of the virtual host computer substantially indicates the processing ability of the web server site (the site in which the web server operates).

3) Moreover, in the virtual host computer system, the access is accepted in the server site in which the virtual host computer is present, the access of the web server in its own server site is leveled, and the respective virtual host computers are necessary for the respective web server sites provided with the dispersed/arranged web servers. Moreover, predetermined different URLs need to be set in the respective virtual host computers. Therefore, a problem occurs that the accessing client has to select the dispersed/arranged web server site by itself.

4) Furthermore, since the accessing client selects the web server site, a problem occurs that the accesses to the respective web server sites cannot be leveled.

However, the transfer system by the TCP connection hop as shown in Fig. 7 has an advantage that, different from the virtual host computer system, a mechanism for temporarily accepting the exclusive access is not separately necessary.

5) However, the accesses are concentrated on the web server in which the scheduler is permanently stationed. Therefore, if a trouble is generated in the web server in which the scheduler is permanently stationed, a problem occurs that the entire server site cannot function.

6) Moreover, in this system, the access to the dispersed/arranged web server site cannot be controlled to the optimum. Since the system is constituted for a purpose of leveling the access in the web server site (in its own

network) managed by the scheduler, predetermined different URLs need to be set in the respective web server sites similarly as the above-described virtual site system. Therefore, there is a problem that the accessing client selects the dispersed/arranged web server site by itself.

7) Furthermore, similarly as the above-described virtual site system, since the accessing client selects the web server site, the accesses to the respective web server sites cannot be leveled.

SUMMARY OF THE INVENTION

[0006] The present invention is constituted by a network status server comprising: logical distance obtaining means for obtaining respective logical distances between respective sites provided with dispersed/arranged information distribution servers and an accessed client; and site determination means for determining an optimum site from the respective sites provided with the dispersed/arranged information distribution servers based on the obtained logical distances.

[0007] Moreover, the present invention is constituted by a network status server comprising: logical distance obtaining means for obtaining respective logical distances between respective sites provided with dispersed/arranged information distribution servers and an accessed client; collection means for collecting network state information between the respective sites provided with the dispersed/arranged information distribution servers and the accessed client, and state information in the site provided with the dispersed/arranged information distribution servers; and server determination means for determining the optimum information distribution server from the dispersed/arranged information distribution servers based on the logical distance obtained by the logical distance obtaining means, the network state information between the respective site provided with the dispersed/arranged information distribution servers and the accessed client, and the state information in the sites provided with the dispersed/arranged information distribution servers collected by the collection means.

[0008] Furthermore, the present invention is constituted by a network status server comprising: logical distance obtaining means for obtaining respective logical distances between respective sites provided with dispersed/arranged information distribution servers and an accessed client; collection means for collecting network state information between the respective sites provided with the dispersed/arranged information distribution servers and the accessed client, and state information in the site provided with the dispersed/arranged information distribution servers; site determination means for, when it is judged that the access from the client is a first access, determining an optimum site from the respective sites provided with the dispersed/arranged information distribution servers based on the logical distance obtained by the logical distance obtaining means; first server determination means for determining an optimum information distribution server from the information distribution servers in the optimum site determined by the site determination means based on a predetermined condition; and second server determination means for, when it is judged that the access from the client is not the first access, determining the optimum information distribution server from the dispersed/arranged information distribution servers based on the logical distance obtained by the logical distance obtaining means, the network state information between the respective sites provided with the dispersed/arranged information distribution servers and the accessed client, and the state information in the site provided with the dispersed/arranged information distribution servers collected by the collection means.

[0009] Moreover, according to the present invention, there is provided an information distribution system comprising: dispersed/arranged information distribution servers; and a network status server.

[0010] The information distribution server comprises: inquiry means for inquiring of the network status server about the optimum information distribution server among the dispersed/arranged information distribution servers in response to an accessing client; response reception means for receiving a response from the network status server; and indication means for indicating the optimum information distribution server to the accessing client. The network status server comprises: inquiry reception means for receiving the inquiry from the inquiry means; logical distance obtaining means for obtaining respective logical distances between respective sites provided with the dispersed/arranged information distribution servers and the accessed client; site determination means for determining the optimum site from the respective sites provided with the dispersed/arranged information distribution servers based on the obtained logical distance; server determination means for determining the optimum information distribution server from the information distribution servers in the optimum site determined by the site determination means based on a predetermined condition; and response transmission means for returning the server determined by the server determination means as the optimum server to the information distribution server having transmitted the inquiry.

[0011] Furthermore, according to the present invention, there is provided an information distribution system comprising: dispersed/arranged information distribution servers; and a network status server.

[0012] The information distribution server comprises: inquiry means for inquiring of the network status server about the optimum information distribution server among the dispersed/arranged information distribution servers in response to an accessing client; response reception means for receiving a response from the network status server; and indication means for indicating the optimum information distribution server to the accessing client. The network status

server comprises: inquiry reception means for receiving the inquiry from the inquiry means; logical distance obtaining means for obtaining respective logical distances between respective sites provided with the dispersed/arranged information distribution servers and the accessed client; collection means for collecting network state information between the respective sites provided with the dispersed/arranged information distribution servers and the accessed client, and state information in the site provided with the dispersed/arranged information distribution servers; server determination means for determining the optimum information distribution server from the dispersed/arranged information distribution servers based on the logical distance obtained by the logical distance obtaining means, the network state information between the respective sites provided with the dispersed/arranged information distribution servers and the accessed client, and the state information in the site provided with the dispersed/arranged information distribution servers collected by the collection means; and response transmission means for returning the server determined by the server determination means as the optimum server to the information distribution server having transmitted the inquiry.

[0013] Additionally, according to the present invention, there is provided an information distribution system comprising: dispersed/arranged information distribution servers; and a network status server.

[0014] The information distribution server comprises: inquiry means for inquiring of the network status server about the optimum information distribution server among the dispersed/arranged information distribution servers in response to an accessing client; response reception means for receiving a response from the network status server; and indication means for indicating the optimum information distribution server to the accessing client. The network status server comprises: inquiry reception means for receiving the inquiry from the inquiry means; logical distance obtaining means for obtaining respective logical distances between respective sites provided with the dispersed/arranged information distribution servers and the client accessing the information distribution server; collection means for collecting network state information between the respective sites provided with the dispersed/arranged information distribution servers and the accessed client, and state information in the site provided with the dispersed/arranged information distribution servers; site determination means for, when it is judged that the access from the client is a first access, determining the optimum site from the respective sites provided with the dispersed/arranged information distribution servers based on the logical distance obtained by the logical distance obtaining means; first server determination means for determining the optimum information distribution server from the information distribution servers in the optimum site determined by the site determination means based on a predetermined condition; second server determination means for, when it is judged that the access from the client is not the first access, determining the optimum information distribution server from the dispersed/arranged information distribution servers based on the logical distance obtained by the logical distance obtaining means, the information of the network state between the respective sites provided with the dispersed/arranged information distribution servers and the accessed client collected by the collection means, and the state information in the site provided with the dispersed/arranged information distribution servers; and response transmission means for returning the server determined by the first server determination means or the second server determination means as the optimum server to the information distribution server having transmitted the inquiry.

[0015] The present invention relates to the server, the system, a control method of controlling the server and system, or a storage medium for storing a control program for controlling the server and system.

[0016] Since the present invention is constituted as described above, the access load between the dispersed/arranged web server sites can be leveled.

[0017] Moreover, the access between the dispersed/arranged web server sites is leveled, while the access load between the web servers in each web server site can be leveled.

[0018] Furthermore, in order to determine the optimum web server at a high speed, the optimum web server site is first determined at a high speed by a first determination method, and the true optimum web server is then determined by a second determination method, so that the optimum web server can be determined at the speed as high as possible.

[0019] Additionally, in order to determine the optimum web server at a high speed, the accessed web server can detect the network state and server state at a high speed.

[0020] Moreover, the access client can be switched to the optimum web server without requiring any active operation.

[0021] Furthermore, when all the dispersed/arranged web servers accept the first access, the concentration of the accesses can be eliminated.

[0022] Other features and advantages of the patent invention will be apparent from the following description in conjunction with the accompanying drawings, in which like reference characters designate the same or similar parts throughout the figures thereof.

BRIEF DESCRIPTION OF THE DRAWINGS

[0023] The accompanying drawings, which are incorporated in and constitute a part of the specification, illustrate embodiments of the invention and together with the description, server to explain the principles of the invention.

[0024] Fig. 1 is an explanatory view of a constitution according to an embodiment of the present invention.

[0025] Fig. 2 is an explanatory view of a function and operation according to the embodiment of the present invention.

[0026] Figs. 3A, 3B, 3C, 3D and 3E are flowcharts showing the function and operation according to the embodiment of the present invention.

[0027] Fig. 4 is an explanatory view showing the function and operation of an NS server according to the embodiment of the present invention.

[0028] Fig. 5 is an explanatory view showing the function and role according to the embodiment of the present invention.

[0029] Fig. 6 is an explanatory view of a conventional virtual host computer system.

[0030] Fig. 7 is an explanatory view of a conventional TCP connection hop system.

[0031] Fig. 8 is a diagram showing an approximation curve during approximation of collected information with a predetermined function.

[0032] Fig. 9 is a block diagram of a computer which can be realized by an NS server, information distribution server and route server of the present invention.

[0033] Fig. 10 is a schematic view showing a storage medium for storing a program code and a transmission device for transmitting the program code.

[0034] Fig. 11 is an image diagram showing the memory map of the storage medium for storing a network status server control program or an information distribution system control program in the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0035] An embodiment of the present invention will be described in detail hereinafter with reference to the drawings. Fig. 1 is a constitution system diagram of an optimum server selection system of the present invention. Shown is an example in which Internet is used as communication means between respective sites. In Fig. 1, a terminal side site 1 accesses an information distributing server, and a server site 2 is provided with an accessed information distributing server (hereinafter referred to as the information distribution server or the web server). Here, the site means the same logical network block, and indicates, for example, the address block of class B or C as a network address class. Of course, even when the block is physically in the same place, it can be regarded as the site. Moreover, in the present embodiment, the server site 2 is dispersed and disposed to perform high-speed large-capacity information distribution. Fig. 1 shows dispersed/arranged server sites 12. Numeral 3 denotes Internet as communication means among the respective sites. In the present embodiment, the communication means among the sites is Internet, but is not limited to this, and for example, a packet communication network, a public network (telephone network), and the like may be used.

[0036] A terminal 4 accesses the information distribution server, and is an information processor in which a standard browser accessible to the information distribution server (web server) is installed in the present embodiment. A terminal side router 5 is a router for performing the path control of the inside and outside of an access side site. A site side router 6 is a router for performing the path control of the inside and outside of the site provided with the information distribution server.

[0037] In an information distribution server 7, a web server software 8 as a standard information distributing software and an agent software 9 newly prepared in the present invention operate. The web server software 8 is a server software provided with a function of transferring information written in html language in response to a request from the outside. Moreover, the agent software 9 is a software provided with various server operation situations and indications for the web server software 8. Additionally, needless to say, the information distribution server may operate with the software integrally formed by the web server software 8 and agent software 9.

[0038] The respective site side routers 6 are provided with the path information obtained when the respective sites view the other sites through Internet (network). A route server 10 is provided with a function of obtaining path information (e.g., BGP information, RIP information, OSPF information, and the like) from the respective site side routers 6. Moreover, the route server 10 has a function of transmitting the path information to a network status server (hereinafter referred to as the NS server) 11.

[0039] The NS server 11 instructs the agent 9 to measure various operation states of the site 2 provided with the dispersed/arranged information distribution servers and the information distribution servers in the site 12, and obtains and stores measurement results. Moreover, the agent 9 measures the network states (e.g., congestion degree, router step number, packet loss ratio, response time, and the like) between terminal side site 1 having requested for the information and the respective information distribution servers 2 and 12, and the NS server 11 obtains and stores the network states measured by the agent 9. Furthermore, the NS server 11 obtains the information from the route server 10 to obtain the network distance between the terminal side site 1 having requested for the information and the sites 2 and 12 provided with the respective information distribution servers.

[0040] Additionally, in Fig. 1, the route server 10 and NS server 11 are in the site 2, but are not limited in the site 2, and may be in any other site.

[0041] An operation for controlling the access to the optimum web server in the present embodiment will next be described. First, Fig. 2 is a schematic view of the web server access. Here, an operation performed when the terminal 4 of the site 1 accesses the web server is concretely shown. Shown is a case in which the terminal 4 in the autonomous (AS) system 1 accesses the web server provided with the access address (hereinafter referred to as URL) of the web site of www.abcde.jp. First, in order to obtain the IP address of the target web server, a nearest domain name server (DNS) in its network (site 1) is inquired from the URL, and the corresponding IP address (addrl) is received (201). Subsequently, the desired web site (the web site having the URL of www.abcde.jp) is accessed (= requested) by the IP address (addrl) (202). When the web server A having the IP address (addrl) accepts the access, the agent 9 inquires of the network status server (NS server) whether or not the web server is an optimum server for the accessed client (203). The NS server inquires of the route server 10 about the network distance between the respective sites 2, 3, 4 provided with the dispersed/arranged web servers and the accessed client site 1 (204). Since the route server 10 exchanges the path information with the border gateway router 6 in each site, the distance between each site and the client site 1 can quickly be returned (205). The NS server 11 receives the response from the route server 10, determines the site 3 as the closest site in the network, and returns the URL of the optimum web server B to the agent 9 of the site 4 (206). Alternatively, the IP address (addr2) of the optimum web server may be returned. In the present embodiment, the AS path information of the border gateway protocol (BGP) is used as a site selection method at this time (when the client first accesses the site) so that the information can quickly be obtained. For this the site 3 is selected as the site with a short AS path from each web server to the client site, and a predetermined method is used to select the web server B having addr2 in the site 3 as the optimum web server.

[0042] Additionally, in addition to the path information, the NS server operates to obtain number of routers between each site and the client (router hop number), response time, network information such as a packet loss ratio, network information in each site (congestion degree, number of packets, number of packet errors, and the like), server load situation (CPU load ratio, CPU idle ratio, number of connection links, disk load ratio, and the like), and other information. The NS server prepares for the next inquiry and stores the information (including temporary storage) (207).

[0043] The agent 9 in the web server A instructs the client 4 to access the web server B during the next access (208). For example, the web server to be connected to the client can automatically be changed using the location command of the HTML language. As a result, the client 4 will automatically access the optimum web server B from the next access.

[0044] An operation performed when the client 4 continuously accesses the web server B will next be described similarly with reference to Fig. 2. The web server B is accessed through the above-described process (209). When the web server B accepts the access, the agent 9 inquires of the NS server 11 whether or not the web server itself is an optimum server for the accessed client (210). The NS server 11 collects the path information and integrates various state information similarly as the operation during the above-described first access and determines the optimum server site and optimum web server. In addition to the path information obtained and stored by the previous access, the NS server 11 obtains number of routers between each site and the client (number of router hops), response time, network information such as the packet loss ratio, network information in each site (congestion degree, number of packets, number of packet errors, and the like), and server state information (CPU load ratio, CPU idle ratio, number of connection links, disk load ratio, and the like) which could not be inspected during the previous access, and the optimum web server to be accessed by the client is calculated and determined from the obtained various information. At this time the selection method comprises applying a numeric weight to each information in accordance with the content, and comparing the total weight for each site or each server to determine the optimum web server C (211). As a result, the client 4 will automatically access the optimum web server C from the next access (214).

[0045] Furthermore, when the client 4 successively accesses the web server C, the operation similar to the above-described operation of accessing the web server B is performed (215, 216), but the operation of accessing the NS server is performed after a predetermined time. Subsequently, when a new optimum web server is determined by the obtained and stored various network state information and server state information in the predetermined time, the client 4 changes the access to the new web server.

[0046] An operation of controlling the access to the optimum web server of the present embodiment will next be described with reference to Figs. 3A to 3E. Fig. 3A is a flowchart showing an operation performed from when the access client first accesses the web server A until the access is changed to the first optimum web server B. Fig. 3B is a flowchart showing the operation of the NS server. Fig. 3C is a flowchart showing an operation performed from when the access client next accesses the web server B until the access is changed to the optimum access web server C. Fig. 3D is a flowchart showing the operation of the NS server. Fig. 3E is a flowchart showing an operation performed from when the access client accesses the web server C and further inspects the optimum access web server until the web server C is again accessed.

[0047] The operation will first be described with reference to Figs. 3A, 3B. In order to obtain the IP address of the target web server from the URL, the closest domain name server (hereinafter abbreviated as DNS) in its network (site 1) is inquired, and the corresponding IP address (addr1) is received (S1).

[0048] Subsequently, the desired web site A is accessed by the IP address (addr1) (S2).

[0049] When the web server A accepts the access, the agent 9 inquires of the NS server whether or not the web server is an optimum server for the accessed client (S3).

[0050] In the NS server, the network distance from the dispersed/arranged respective web server sites to the accessed client site 1 is inquired of the route server 10 (S4).

[0051] Since the route server 10 exchanges the path information with the site side router 6 in each site (hereinafter referred to as the border gateway router), the distance from each site to the client site 1 can quickly be returned, and the NS server obtains the path information and distance (S5).

[0052] The NS server 11 receives the response from the route server 10, fixes the site 3 as the close site in the network, determines the optimum web server B in the site 3 in a predetermined method, and answers the agent 9 of the site 4 that the optimum web server is the web server B (S6). Additionally, the predetermined method of fixing the optimum web server in S6 may comprise predetermining the web server for each site, determining the web server in a predetermined order for each site, or arbitrarily determining the web server in the site. Moreover, since the NS server obtains the state information in the site provided with the web server (e.g., the information of the network state in the site such as the congestion degree, number of packets and packet errors, and the server state information such as CPU load ratio, CPU idle ratio, number of connection links and disk load ratios), the optimum web server may be determined utilizing the state information in the site.

[0053] The NS server stores the obtained path information (including the temporary storage) to prepare for the next inquiry (S7).

[0054] On the other hand, the agent 9 in the web server A answers the client 4 that the optimum web server B is accessed from the next access (S8).

[0055] As a result, the client 4 will automatically access the optimum web server B from the next access.

[0056] The operation will next be described with reference to Figs. 3C, 3D.

[0057] The web server B is accessed through the above-described process (S9).

[0058] When the web server B accepts the access, the agent 9 inquires of the NS server 11 whether or not the web server is an optimum server for the accessed client (S10).

[0059] The NS server 11 collects the path information and integrates various state information similarly as the operation during the above-described first access and determines the optimum server site and optimum web server. In addition to the path information obtained and stored by the previous access, the NS server 11 obtains number of routers between each site and the client (router hop number), response time, network information such as the packet loss ratio, network information in each site (congestion degree, number of packets, packet error, and the like), and web server state information (CPU load ratio, CPU idle ratio, number of connection links, disk load ratio, and the like) which could not be inspected during the previous access, and the optimum web server to be accessed by the client is calculated and determined from the obtained various information. At this time the selection method comprises applying the numeric weight to each information in accordance with the content, and comparing the total weight for each site or each server to determine the optimum web server C (S11).

[0060] The NS server 11 notifies the agent 9 in the web server B of the optimum web server C (S12), and further answers the client 4 that the optimum web server C is accessed from the next access (S13).

[0061] As a result, the client 4 will automatically access the optimum web server C from the next access (S14).

[0062] Finally, the operation will be described with reference to Fig. 3E. When the client 4 successively accesses the web server C, the optimum web server is inquired of the NS server similarly as the above-described access to the web server B (S15), but here the web server C is judged to be the optimum web server similarly as the previous access, and the processing for changing the web server to be accessed (S16) is not performed. However, when it is answered in S15 that the web server different from the web server C is the optimum web server, needless to say the processing for changing the web server to be accessed is performed.

[0063] Additionally, the operation of accessing the NS server is intermittently performed after a predetermined time. Subsequently, when a new optimum web server is determined by the obtained and stored various network state information and server state information in the predetermined time, the client 4 changes the access to the new web server.

[0064] An operation will next be described with reference to Fig. 4, in which the NS server 11 obtains and stores the state of the network from the web server to the client, the state of the network in the web server site, and information on the web server state.

[0065] In the present system, the following two categories of information are periodically (intermittently) collected:

(1) Network state from each server site to the client;
(2) State in each server site (network state, server state).

[0066] Fig. 4 shows a relation. The NS server 11 periodically measures the network state from each server site to the accessed client. Items to be measured are described later.

[0067] Each server site router 6 holds the network path information seen from the server site, and the route server

10 collects the path information owned by the router from each server site router 6. The route server 10 has a logical connection as being peer-to-peer with each server site router 6, that is, the border gateway router (BGR) or an external network border router (401). Subsequently, the analysis result of the predetermined path information is obtained in response to the request from the NS server 11, and returned to the NS server 11 (402). For other inspection items, the NS server autonomously performs the inspection (403), but in some cases the NS server 11 instructs the agent software 9 incorporated in each web server software to perform inspection in accordance with the inspection content.

[0068] For the network and server states in each server site, the agent software 9 incorporated in the web server software of each server site periodically performs the inspection in accordance with the instruction from the NS server 11. The collected various information are stored in the NS server in time series. In this data storage method, time is intermittently stored as information attached to the various information. In the present embodiment, various information are inspected, obtained, and stored every 30 minutes. Moreover, it is empirically known that the stored information periodically changes in predetermined periods such as a day unit, week unit, month unit, and year unit. Therefore, in the present system, the stored information is approximated with a predetermined function.

[0069] The time unit for the approximation may be any one of the day unit, week unit, month unit, and year unit, but in the present embodiment, various information are approximated by the day unit by a three-dimensional function and a coefficient is saved. For example, when the congestion degree (CSn) of the network in the server site is approximated with the three-dimensional function, a time axis is set to t, and the approximation is performed with an approximation curve shown in Fig. 8. When the approximation equation is represented as $CSn = a \cdot t^3 + b \cdot t^2 + c \cdot t + d$ (a, b, c, d being coefficients), the NS server saves this coefficient (the three-dimensional function itself may be saved).

[0070] As a result, when the request for selecting the optimum server is received from each server, by utilizing the three-dimensional function, various states are momentarily estimated, and the optimum server is selected based on the estimated value.

[0071] The collection of the network state information between each server site and the access client is not infinitely continued, and is continued while the access to the dispersed/arranged servers is continued and for a predetermined period after that. In the present embodiment, the period is set to one week. When there is no access from the access client 4 for one week, the information collecting operation to the access client 4 from each site is not performed.

[0072] Moreover, when there is no access from the access client 4 for the predetermined period (e.g., three weeks), the network state information (the approximation equation, and the like) between the access client 4 and each server site stored in the NS server stored in the NS server may be discarded. In this case, no wasteful information is stored, and the storage capacity necessary for the NS server can advantageously be saved.

[0073] Furthermore, when there is no access from the access client 4 for the predetermined period, the path information between the access client 4 and each server site stored in the NS server may also be discarded.

[0074] Additionally, the stored information can also be treated as the result representative of a specific area. Specifically, the NS server obtains and stores the path information (logical distance) between each server site and the access client or the network state information between each server site and the access client. For the path information (logical distance) between each server site and the access client, the path information (logical distance) between each server site and the site to which the access client belongs may be utilized. For the network state information between each server site and the access client, the network state information between each server site and the access client may be utilized.

[0075] For example, when the IP address of the access client 4 is 192.168.0.1, the network state with this client is used as the result of the entire class C address block. Specifically, for the access from the network block of 192.168.0.0/24, the already inspected and obtained information of 192.168.0.1 is used. In this case, the memory space for the data to be stored can be reduced. In the present embodiment, the access client area is classified with the class B access (the network address block corresponding to A.B.0.0/16).

[0076] The network and information distribution server state information autonomously inspected and obtained by the NS server 11 will next be described.

(1) The network state information from each information distribution server site to the client are as follows:

- ASLn: distance between the logical networks by AS path (BGP path information is used);
- RTn: response time;
- RNn: number of router steps (number of router hops); and
- PLn: packet loss ratio.

(2) The state (network, server) information in each information distribution server site are as follows:
as the network state information in the site,

- CSn: congestion degree in the site;

- PSn: number of packets in the site; and
- ESn: packet error,

as the information distribution server state information,

- CPUnm: CPU load ratio;
- IDLEnm: CPU idle value;
- LINKnm: number of connection links; and
- IOnm: disk load ratio,

additionally, n: server site number, m: server number.

[0077] A method of using these network and server state information to determine the optimum server. The NS server 11 stores these various state information, and uses these information to determine the optimum server site and server. In the present embodiment, these are calculated using the following equation.

[0078] A selection method I corresponds to the selection method in the step S6 in Fig. 3B, and the site in which ASLn is minimum is fixed as the optimum site.

[0079] A selection method II corresponds to the selection method in the step S11 in Fig. 3D.

[0080] A network state value K1n between the server site and the client, a network state value K2n in the server site, and a server state value are obtained, and an optimum site judgment value Kn is obtained for each server site as shown by the following equation. The site number (n) in which the value is minimum is fixed as the optimum site. Subsequently, an optimum server judgment value Snm is obtained using K3nm. The optimum server judgment value is obtained from the optimum site judgment value and server state value. Therefore, by using the network state of the client site and server site, network state in the site, server state, and logical distance between the networks with weight coefficients A to O, the optimum server is generally judged. Additionally, here, the server in which Snm is minimum is fixed as the optimum server.

Network state value:

$$K1n = RTn \cdot A + RNn \cdot B + PLn \cdot C$$

In-site network state value:

$$K2n = Csn \cdot D + PSn \cdot E + ESn \cdot F$$

Server state value:

$$K3nm = CPUnm \cdot G + IDLEnm \cdot H + LINKnm \cdot I + IOnm \cdot J$$

Optimum site judgment value:

$$Kn = K1n \cdot K + K2n \cdot L + ASLn \cdot M$$

Optimum server judgment value:

$$Snm = Kn \cdot N + K3nm \cdot O$$

[0081] Additionally, n: number of server sites, m: number of servers, A to O: weight coefficients.

[0082] Moreover, in the present embodiment, the response time, number of router steps (number of router hops), and packet loss ratio are used as the network state value, the congestion degree in the site, number of packets in the site, and packet error are used as the in-site network state value, and CPU load ratio, CPU idle value, number of connection links, and disk load ratio are used as the server state value, but with respect to the network state value, in-site network state value and server state value, at least one of the utilized information in the present embodiment may be utilized.

[0083] Moreover, the network state information in the site and the state information of the information distribution server are utilized as the state information in the server site, but either one may be used as the representative during the determination of the optimum server.

[0084] The function of the route server used in the present system will next be described with reference to Fig. 5. The current Internet/Intranet is said to be an environment in which routers are connected in a mesh manner. In the system, in order to distribute the desired packet to a specific place, a routing protocol is used, and the routing protocol called Border Gateway Protocol (BGP) is used in a wide network. In the BGP, each router is provided with a path table, and the packet is distributed to the specific place based on this path table.

[0085] In Fig. 5, the route server 10, each site router and BGP are logically connected as being peer-to-peer. Here, as one technique of measuring the distance between each site and the client, AS-path number is utilized. This AS-path number means the number of independent networks treated in the BGP, and the path information up to the desired AS number are mapped as the connection of AS numbers in the path information. Specifically, the path information between the server site and the client site can be known as the connection of AS numbers applied to the independent networks. The AS-path number indicates the frequency at which the independent networks are passed, and can be known from the connection of the AS numbers.

[0086] The route server can collect the path information viewed from each server site by obtaining the path information from the router 6 in each server site, and the server site closest to the access client (in the AS path) can be selected.

[0087] In the present embodiment, this route server is used to obtain the BGP distance between the access client and each server site, that is, the AS-path number, and the distance is used as one of the selection conditions for determining the optimum server site.

[0088] Moreover, in the present embodiment, used are the selection method I of obtaining the optimum site from the logical distance from the server site to the client and then determining the information distribution server in the optimum site in the predetermined method, and the selection method II of determining the optimum information distribution server from the logical distance, network state information and information distribution server state information. For the first access from the client, the selection method I is used to determine the optimum server, and for the second and subsequent accesses, the selection method II is used to determine the optimum server, but the present invention is not limited to this order, and the selection method I may constantly be used, or the selection method II may be used.

[0089] Moreover, when the client accesses the NS server, and the path information (logical distance) between the client and the server site is stored, the stored path information may be utilized to determine the optimum server.

[0090] Furthermore, the NS server, information distribution server and route server of the present invention can be realized by the information processor with a constitution equal to that of the computer shown in Fig. 9. A computer 901 of Fig. 9 is constituted of a CPU 902 for reading a program to perform an actual processing, RAM 903 used by the CPU 902 during the processing, hard disk (hereinafter referred to as HD) 904 as the storage medium for supplying a program code, floppy disk (FD) 905, communication interface 906 used during connection to the network, and the like. The network status server control program and information distribution system control program of the present invention may be performed by the computer with the constitution equal to that of Fig. 9 by the program installed from the outside. In this case, as shown in Fig. 10, needless to say, the object of the present invention is achieved by providing the system or the device with a recording medium 1002 in which the software program code for realizing the function of the above-described embodiment is recorded, and reading and executing the program code stored in the storage medium by the computer (or CPU or MPU) of the system or the device.

[0091] In this case, the program code itself read from the storage medium realizes the inventive function of the present invention, and the storage medium for storing the program code constitutes the present invention.

[0092] As the storage medium for supplying the program code, for example, magnetic disks such as a floppy disk and hard disk, optical disks, optical magnetic disks, CD-ROM, CD-R, DVD-ROM, DVD-RAM, magnetic tapes, memory cards, ROM, and the like can be used.

[0093] Moreover, it goes without saying that the present invention can be applied to a case in which the program is distributed from the storage medium for recording the program code of the software for realizing the function of the above-described embodiment via the communication line of the personal computer communication, and the like.

[0094] Fig. 11 is a diagram showing one example of the memory map of storage media such as the CD-ROM for supplying the program code. An area 1101 for storing directory information indicates the positions of an area 1102 for storing an install program, and an area 1103 for storing control programs such as a network status server control program and an information distribution system control program. When the network status server control program or the information distribution system control program of the present invention is installed in the computer, the install program stored in the area 1102 is loaded to the system, and executed by the CPU. Subsequently, the install program executed by the CPU reads the network status server control program or the information distribution system control program from the area 1103 for storing the control programs such as the network status server control program and information distribution system control program and stores the program in the hard disk.

[0095] When the present invention is applied to the storage medium, the storage medium stores the program code

corresponding to the flowchart described above.

[0096] Moreover, it goes without saying that, as shown in Fig. 10, the object of the present invention is achieved by transmitting the program code of the software for realizing various functions in the above-described embodiment from a transmission device 1004, for example HTTP server or FTP server, receiving the code via a network 1005, public line or radio, and executing the program code by the computer (or CPU, MPU) of the system or device.

[0097] In this case, since the program code itself transmitted from the transmission device realizes the function in the above-described embodiment, the transmission device for transmitting the program code constitutes the present invention.

[0098] Moreover, the present invention can be applied not only to the realization of the function of the above-described embodiment by executing the program code read by the computer but also to the realization of the function of the above-described embodiment by performing a part or the whole of the actual processing by the OS, and the like operating on the computer based on the instruction of the program code.

[0099] As described above, according to the present embodiment, there is an effect that the access load between the dispersed/arranged web server sites can be leveled.

[0100] Moreover, the access between the dispersed/arranged web server sites is leveled, and the access load can effectively be leveled even between the web servers in each web server site.

[0101] Furthermore, in order to determine the optimum web server at a high speed, when the client has a first access, the optimum web server site is determined using the path information (logical distance) at a high speed in the first determination method. In the subsequent access from the client, the optimum web server can be determined at a high speed by using not only the path information but also the network state information and the state information in the web server site in the second determination method to determine the optimum web server, which produces an effect that the web server can respond to the access client at a high speed.

[0102] Additionally, since the network state information and server state information necessary for recognizing the optimum web server are automatically obtained and stored in the NS server, or the NS server obtains and stores the information, and the optimum web server is generally determined by the information, there is an effect that the accessed web server can determine the optimum web server at a high speed.

[0103] Moreover, the client is switched to the optimum web server without adding the separate device or requiring any active operation, and all the web servers in the dispersed/arranged web server sites can accept the access, the concentration of the accesses can effectively be eliminated.

[0104] Further, the computer program for carrying out the above disclosed method can be obtained in electronic form for example by downloading the code over a network such as the internet. Thus in accordance with another aspect of the present invention there is provided an electrical signal carrying processor implementable instructions for controlling a processor to carry out the method as hereinbefore described.

Claims

1. A network status server comprising:

   logical distance obtaining means for obtaining respective logical distances between respective sites provided with dispersed/arranged information distribution servers and an accessed client; and
   site determination means for determining an optimum site from said respective sites based on said logical distance.

2. A network status server comprising:

   logical distance obtaining means for obtaining respective logical distances between respective sites provided with dispersed/arranged information distribution servers and an accessed client;
   collection means for collecting network state information with said client, and state information in said respective sites; and
   server determination means for determining the optimum information distribution server from said dispersed/arranged information distribution servers based on the logical distance obtained by said logical distance obtaining means, the network state information between said respective sites and said client, and the state information in said site collected by said collection means.

3. A network status server comprising:

   logical distance obtaining means for obtaining respective logical distances between respective sites provided

with dispersed/arranged information distribution servers and an accessed client;
collection means for collecting network state information between said respective sites and said client, and state information in said site;
site determination means for, when it is judged that the access from said client is a first access, determining the optimum site from the respective sites based on the logical distance obtained by said logical distance obtaining means;
first server determination means for determining the optimum information distribution server from the information distribution servers in the optimum site determined by said site determination means based on a predetermined condition; and
second server determination means for, when it is judged that the access from the client is not the first access, determining the optimum information distribution server from said dispersed/arranged information distribution servers based on the logical distance obtained from said logical distance obtaining means, the network state information between said respective sites and said client, and the state information in said site collected by said collection means.

4. An information distribution system comprising: dispersed/arranged information distribution servers; and a network status server,

said information distribution server comprising:

inquiry means for inquiring of said network status server about the optimum information distribution server among said dispersed/arranged information distribution servers in response to an accessing client;
response reception means for receiving a response from said network status server; and
indication means for indicating said optimum information distribution server to said accessing client,

said network status server comprising:

inquiry reception means for receiving the inquiry from said inquiry means;
logical distance obtaining means for obtaining respective distances between said respective sites provided with the dispersed/arranged information distribution servers and said client;
site determination means for determining the optimum site from the respective sites based on said obtained logical distance;
server determination means for determining the optimum information distribution server from the information distribution servers in the optimum site determined by said site determination means based on a predetermined condition; and
response transmission means for returning the server determined by said server determination means as the optimum server to said dispersed/arranged information distribution servers having transmitted the inquiry.

5. An information distribution system comprising: dispersed/arranged information distribution servers; and a network status server,

said information distribution server comprising:
inquiry means for inquirihg of said network status server about the optimum information distribution server among said dispersed/arranged information distribution servers in response to an accessing client;
response reception means for receiving a response from said network status server; and
indication means for indicating said optimum information distribution server to said accessing client,

said network status server comprising:

inquiry reception means for receiving the inquiry from said inquiry means;
logical distance obtaining means for obtaining respective distances between respective sites provided with said dispersed/arranged information distribution servers and said client;
collection means for collecting network state information between said respective sites and said client, and state information in said site;
server determination means for determining the optimum information distribution server from said dispersed/arranged information distribution servers based on the logical distance obtained by said logical distance obtaining means, the network state information between said respective sites and said client, and said state information in the site collected by said collection means; and
response transmission means for returning the server determined by said server determination means as said

optimum server to said dispersed/arranged information distribution servers having transmitted the inquiry.

6. An information distribution system comprising: dispersed/arranged information distribution servers; and a network status server,

said information distribution server comprising:

inquiry means for inquiring of said network status server about the optimum information distribution server among said dispersed/arranged information distribution servers in response to an accessing client;
response reception means for receiving a response from said network status server; and
indication means for indicating the optimum information distribution server to said client,

said network status server comprising:

inquiry reception means for receiving the inquiry from said inquiry means;
logical distance obtaining means for obtaining respective distances between respective sites provided with said dispersed/arranged information distribution servers and the accessing client;
collection means for collecting network state information between said respective sites and said client, and state information in the respective sites;
site determination means for, when it is judged that the access from said client is a first access, determining the optimum site from said respective sites based on the logical distance obtained by said logical distance obtaining means;
first server determination means for determining the optimum information distribution server from the information distribution servers in said respective sites based on a predetermined condition;
second server determination means for, when it is judged that the access from said client is not the first access, determining the optimum information distribution server from said dispersed/arranged information distribution servers based on the logical distance obtained from said logical distance obtaining means, the network state information between said respective sites and said client, and the state information in said respective sites collected by said collection means; and
response transmission means for returning the server determined by said first server determination means or said second server determination means as said optimum server to said dispersed/arranged information distribution servers having transmitted the inquiry.

7. A network status server control method of controlling a network status server, the method comprising:

a logical distance obtaining step of obtaining respective logical distances between respective sites provided with dispersed/arranged information distribution servers and an accessed client; and
site determining step of determining an optimum site from said respective sites based on said obtained logical distance.

8. The network status server control method according to claim 7 wherein said logical distance obtaining step comprises obtaining said logical distance from a route server comprising means for obtaining the logical distance to a predetermined network address from path information between the sites.

9. The network status server control method according to claim 8 wherein said network status server comprises path information storage means,

said network status server control method further comprises a path information storing step of storing the path information between said respective sites and said client, and
when said path information is stored in said path information storage means, said logical distance obtaining step comprises obtaining said logical distance from the path information stored in said path information storage means.

10. The network status server control method according to claim 9 wherein when there is no access from said client for a predetermined period, said path information storing step comprises discarding the path information between said respective sites and the client stored in said path information storage means.

11. The network status server control method according to claim 7 wherein said site determining step comprises determining the site in which the respective logical distances between said respective sites and said client are min-

imum as said optimum site.

12. The network status server control method according to claim 7, further comprising a server determining step of determining the optimum information distribution server from the information distribution servers in the optimum site determined in said site determining step based on a predetermined condition.

13. The network status server control method according to claim 12, further comprising a state information collecting step of collecting state information in the site provided with said dispersed/arranged information distribution servers,
wherein said server determining step comprises determining the optimum information distribution server using the collected state information in said optimum site as a condition.

14. The network status server control method according to claim 13 wherein said network status server comprises state information storage means,

said network status server control method further comprises a state information storing step of storing the collected state information in said site into the state information storage means, and
when the state information in said optimum site is stored in said state information storage means, said server determining step comprises determining the optimum information distribution server using the stored state information in said optimum site as the condition.

15. The network status server control method according to claim 14, further comprising a state information discarding step of discarding the state information in the site stored in said state information storage means after a predetermined period elapses.

16. The network status server control method according to claim 14 wherein said state information storing step comprises approximating and storing said state information in the site by a predetermined approximation equation.

17. The network status server control method according to claim 13 wherein said state information in the site comprises at least one information of said network state information in the site and said state information of the information distribution server in the site.

18. A network status server control method for controlling a network status server, the method comprising:

a logical distance obtaining step of obtaining respective logical distances between respective sites provided with dispersed/arranged information distribution servers and an accessed client;
a collecting step of collecting network state information between said respective sites and said client, and state information in said site; and
a server determining step of determining the optimum information distribution server from said information distribution servers based on said obtained logical distance, the network state information between said respective sites and said client, and the state information in said site collected in said collecting step.

19. The network status server control method according to claim 18 wherein said logical distance obtaining step comprises obtaining said logical distance from a route server comprising means for obtaining the logical distance to a predetermined network address from path information between the sites.

20. The network status server control method according to claim 19 wherein said network status server comprises path information storage means,

said network status server control method further comprises a path information storing step of storing the path information between said respective sites and said client into said path information storage means, and
when said path information is stored in said path information storage means, said logical distance obtaining step comprises obtaining said logical distance from the path information stored in said path information storing step.

21. The network status server control method according to claim 20 wherein when there is no access from said client for a predetermined period, said path information storing step comprises discarding the path information between said respective sites and the client stored in said path information storage means.

22. The network status server control method according to claim 18 wherein said network status server comprises state information storage means,

said network status server control method further comprises a state information storing step of storing the network state information between said respective sites and said client, and the state information in said site into the state information storage means, and
when said state information is stored in said state information storage means, said collecting step comprises collecting said state information from said state information stored in said state information storage means.

23. The network status server control method according to claim 22 wherein when there is no access from said client for a predetermined period, said state information storing step comprises discarding said network state information between the respective sites and the client stored in said state information storage means.

24. The network status server control method according to claim 22, further comprising a state information discarding step of discarding the state information in the site stored in said state information storage means after the predetermined period elapses.

25. The network status server control method according to claim 22 wherein said state information storing step comprises approximating and storing said state information in the site by a predetermined approximation equation.

26. The network status server control method according to claim 18 wherein said state information in the respective sites comprises at least one of said network state information in the respective sites and said state information of the dispersed/arranged information distribution servers.

27. The network status server control method according to claim 18 wherein said network status server comprises path information storage means and state information storage means,

said network status server control method further comprises: a path information storing step of storing path information between said respective sites and said client into the path information storage means; and
a state information storing step of storing network state information between said respective sites and said client, and state information in said respective sites into the state information storage means,
said server determining step comprises determining the information distribution server in which an optimum server judgment value Snm in the following equations satisfies a predetermined condition as the optimum information distribution server based on said stored path information, said state information and a predetermined weight coefficient;

- network state value:

$$K1n = RTn \cdot A + RNn \cdot B + PLn \cdot C$$

- in-site network state value:

$$K2n = Csn \cdot D + PSn \cdot E + ESn \cdot F$$

- server state value:

$$K3nm = CPUnm \cdot G + IDLEnm \cdot H + LINKnm \cdot I + IOnm \cdot J$$

- optimum site judgment value:

$$Kn = K1n \cdot K + K2n \cdot L + AsLn \cdot M$$

- optimum server judgment value: $Snm = Kn \cdot N + K3nm \cdot o$ (additionally, n: number of sites, m: number of servers, A to O: weight coefficients, and for respective symbols, ASLn: logical distance between the net-

works by AS path (using the path information of BGP), RTn: response time, RNn: number of router steps (number of router hops), PLn: packet loss ratio, CSn: congestion degree in the site, PSn: number of packets in the site, ESn: number of packet errors, CPUnm: CPU load ratio, IDLEnm: CPU idle value, LINKnm: number of connection links, IOnm: disk load ratio).

28. A network status server control method comprising:

a logical distance obtaining step of obtaining respective logical distances between respective sites provided with dispersed/arranged information distribution servers and an accessed client;
a collecting step of collecting network state information between said respective sites and said client, and state information in said respective sites;
a site determining step of, when it is judged that the access from said client is a first access, determining the optimum site from said respective sites based on said obtained logical distance;
a first server determining step of determining the optimum information distribution server from the information distribution servers in said optimum site based on a predetermined condition; and
a second server determining step of, when it is judged that the access from said client is not the first access, determining the optimum information distribution server from said dispersed/arranged information distribution servers based on said obtained logical distance, said network state information between said respective sites provided with the dispersed/arranged information distribution servers and said client, and said state information in the site provided with the dispersed/arranged information distribution servers collected in said collecting step.

29. The network status server control method according to claim 28 wherein said logical distance obtaining step comprises obtaining said logical distance from a route server comprising means for obtaining the logical distance to a predetermined network address from path information between the sites.

30. The network status server control method according to claim 29 wherein said network status server comprises path information storage means,

said network status server control method further comprises a path information storing step of storing path information between said respective sites provided with the dispersed/arranged information distribution servers and said client into said path information storage means, and
when said path information is stored in said path information storage means, said logical distance obtaining step comprises obtaining said logical distance from said path information stored in the path information storage means.

31. The network status server control method according to claim 30 wherein when there is no access from said client for a predetermined period, said path information storing step comprises discarding said path information between the respective sites and the client stored in said path information storage means.

32. The network status server control method according to claim 28 wherein said network status server comprises state information storage means,

said network status server control method comprises a state information storing step of storing network state information between said respective sites provided with the dispersed/arranged information distribution servers and said client, and state information in said site provided with the dispersed/arranged information distribution servers into said state information storage means, and
when said state information is stored in said state information storage means, said collecting step comprises collecting said state information from said state information stored in said state information storage means.

33. The network status server control method according to claim 32 wherein when there is no access from said client for a predetermined period, said state information storing step comprises discarding said network state information between the respective sites and the client stored in said state information storage means.

34. The network status server control method according to claim 32, further comprising a state information discarding step of discarding the state information in the site stored in said state information storage means after the predetermined period elapses.

35. The network status server control method according to claim 32 wherein said state information storing step comprises approximating and storing said state information in the site by a predetermined approximation equation.

36. The network status server control method according to claim 28 wherein said network status server comprises path information storage means and state information storage means,

said network status server control method further comprises: a path information storing step of storing path information between said respective sites provided with the dispersed/arranged information distribution servers and said client into said path information storage means; and
a state information storing step of storing network state information between said respective sites provided with the dispersed/arranged information distribution servers and said client, and state information in said site provided with the dispersed/arranged information distribution servers into said state information storage means, and
said second server determining step comprises determining the information distribution server in which an optimum server judgment value Snm in the following equations satisfies a predetermined condition as the optimum information distribution server based on said stored path information, said state information and a predetermined weight coefficient;

- network state value:

$$K1n = RTn \cdot A + RNn \cdot B + PLn \cdot C$$

- in-site network state value:

$$K2n = Csn \cdot D + PSn \cdot E + ESn \cdot F$$

- server state value:

$$K3nm = CPUnm \cdot G + IDLEnm \cdot H + LINKnm \cdot I + IOnm \cdot J$$

- optimum site judgment value:

$$Kn = K1n \cdot K + K2n \cdot L + ASLn \cdot M$$

- optimum server judgment value:

$$Snm = Kn \cdot N + K3nm \cdot O$$

(additionally, n: number of sites, m: number of servers, A to O: weight coefficients, and for respective symbols, ASLn: logical distance between the networks by AS path (using the path information of BGP), RTn: response time, RNn: number of router steps (number of router hops), PLn: packet loss ratio, CSn: congestion degree in the site, PSn: number of packets in the site, ESn: number of packet errors, CPUnm: CPU load ratio, IDLEnm: CPU idle value, LINKnm: number of connection links, and IOnm: disk load ratio).

37. An information distribution system control method for controlling an information distribution system comprising: dispersed/arranged information distribution servers; and a network status server,

said method comprising: in said information distribution server,
an inquiring step of inquiring of said network status server about the optimum information distribution server among said information distribution servers in response to an accessing client;
a response receiving step of receiving a response from said network status server; and
an indicating step of indicating said optimum information distribution server to said accessing client,
in said network status server,

an inquiry receiving step of receiving the inquiry from said inquiring step;
a logical distance obtaining step of obtaining respective distances between said respective sites provided with the dispersed/arranged information distribution servers and said client;
a site determining step of determining the optimum site from said respective sites provided with the dispersed/arranged information distribution servers based on said obtained logical distance;
a server determining step of determining the optimum information distribution server from the information distribution servers in the optimum site determined by said site determining step based on a predetermined condition; and
a response transmitting step of returning the server determined in said server determining step as said optimum server to said information distribution server having transmitted the inquiry.

38. The information distribution system control method according to claim 37 wherein said information distribution system further comprises a route server,

said control method comprises, in said route server a logical distance calculating step of obtaining a logical distance to a predetermined network address from the path information between the sites, and
said logical distance obtaining step comprises obtaining said logical distance calculated in said logical distance calculating step.

39. The information distribution system control method according to claim 37, further comprising: in said information distribution server,

a step of collecting network state information between the site to which the information distribution server belongs and said accessing client; and
a network state transmitting step of transmitting said collected network state information to the network status server.

40. The information distribution system control method according to claim 37, further comprising: in said information distribution server, a step of collecting network state information in the site to which the information distribution server belongs, and the state information of the information distribution server; and
an in-site information transmitting step of transmitting said collected network state information in the site, and the state information of the information distribution server to the network status server.

41. An information distribution system control method for controlling an information distribution system comprising: dispersed/arranged information distribution servers; and a network status server,

said method comprising: in said information distribution server,
an inquiring step of inquiring of said network status server about the optimum information distribution server among said information distribution servers in response to an accessing client;
a response receiving step of receiving a response from said network status server; and
an indicating step of indicating said optimum information distribution server to said client, and
in said network status server,
an inquiry receiving step of receiving the inquiry from said inquiring step;
a logical distance obtaining step of obtaining respective logical distances between said respective sites and said client;
a collecting step of collecting network state information between said respective sites and said client, and state information in said respective sites;
a server determining step of determining the optimum information distribution server from said dispersed/arranged information distribution servers based on the logical distance obtained in said logical distance obtaining step, the network state information between said respective sites and said client, and said state information in the site collected in said collecting step; and
a response transmitting step of returning the server determined in said server determining step as said optimum server to said dispersed/arranged information distribution servers having transmitted the inquiry.

42. The information distribution system control method according to claim 41 wherein said information distribution system comprises a route server,

said control method comprises, in said route server, a logical distance calculating step of obtaining a logical

distance to a predetermined network address from the path information between the sites, and
said logical distance obtaining step comprises obtaining said logical distance calculated in said logical distance calculating step.

43. The information distribution system control method according to claim 41, further comprising: in said information distribution server,

a step of collecting network state information between the site to which the information distribution server belongs and said accessing client; and
a network state transmitting step of transmitting said collected network state information to the network status server.

44. The information distribution system control method according to claim 41, further comprising: in said information distribution server,

a step of collecting network state information of the site to which the information distribution server belongs, and the state information of the information distribution server; and
an in-site information transmitting step of transmitting said collected network state information in the site, and the state information of the information distribution server to the network status server.

45. An information distribution system control method for controlling an information distribution system comprising: dispersed/arranged information distribution servers; and a network status server,

said control method comprising: in said information distribution server,
an inquiring step of inquiring of said network status server about the optimum information distribution server among said information distribution servers in response to an accessing client;
a response receiving step of receiving a response from said network status server; and
an indicating step of indicating said optimum information distribution server to said client,
in said network status server,
an inquiry receiving step of receiving the inquiry from said inquiring step;
a logical distance obtaining step of obtaining respective distances between said respective sites and said client;
a collecting step of collecting network state information between said respective sites and said client, and state information in said respective sites;
a site determining step of, when it is judged that the access from said client is a first access, determining the optimum site from said respective sites provided with the dispersed/arranged information distribution servers based on the logical distance obtained in said logical distance obtaining step;
a first server determining step of determining the optimum information distribution server from the information distribution servers in the optimum site determined in said site determining step based on a predetermined condition;
a second server determining step of, when it is judged that the access from said client is not the first access, determining the optimum information distribution server from said information distribution servers based on the logical distance obtained in said logical distance obtaining step, the network state information between said respective sites provided with the dispersed/arranged information distribution servers and said client, and said state information in the site provided with the dispersed/arranged information distribution servers collected in said collecting step; and
a response transmitting step of returning the server determined in said first server determining step or said second server determining step as said optimum server to said dispersed/arranged information distribution servers having transmitted the inquiry.

46. The information distribution system control method according to claim 45 wherein said information distribution system further comprises a route server,

said control method comprises, in said route server, a logical distance calculating step of obtaining a logical distance to a predetermined network address from the path information between the sites, and
said logical distance obtaining step comprises obtaining said logical distance calculated in said logical distance calculating step.

47. The information distribution system control method according to claim 45, further comprising: in said information

distribution server,

a step of collecting network state information between the site to which the information distribution server belongs and said accessing client; and
a network state transmitting step of transmitting said collected network state information to the network status server.

48. The information distribution system control method according to claim 45, further comprising: in said information distribution server,

a step of collecting network state information of the site to which the information distribution server belongs, and the state information of the information distribution server; and
an in-site information transmitting step of transmitting said collected network state information in the site, and the state information of the information distribution server to the network status server.

49. A computer readable medium storing a computer program element executing any one of methods as claimed in claims 7 to 48.

50. A computer program comprising processor implementable instructions for carrying out a method as claimed in any one of claims 7 to 48.

51. An electrical signal carrying processor implementable instructions for controlling a processor to carry out the method of any one of claims 7 to 48.

FIG. 1

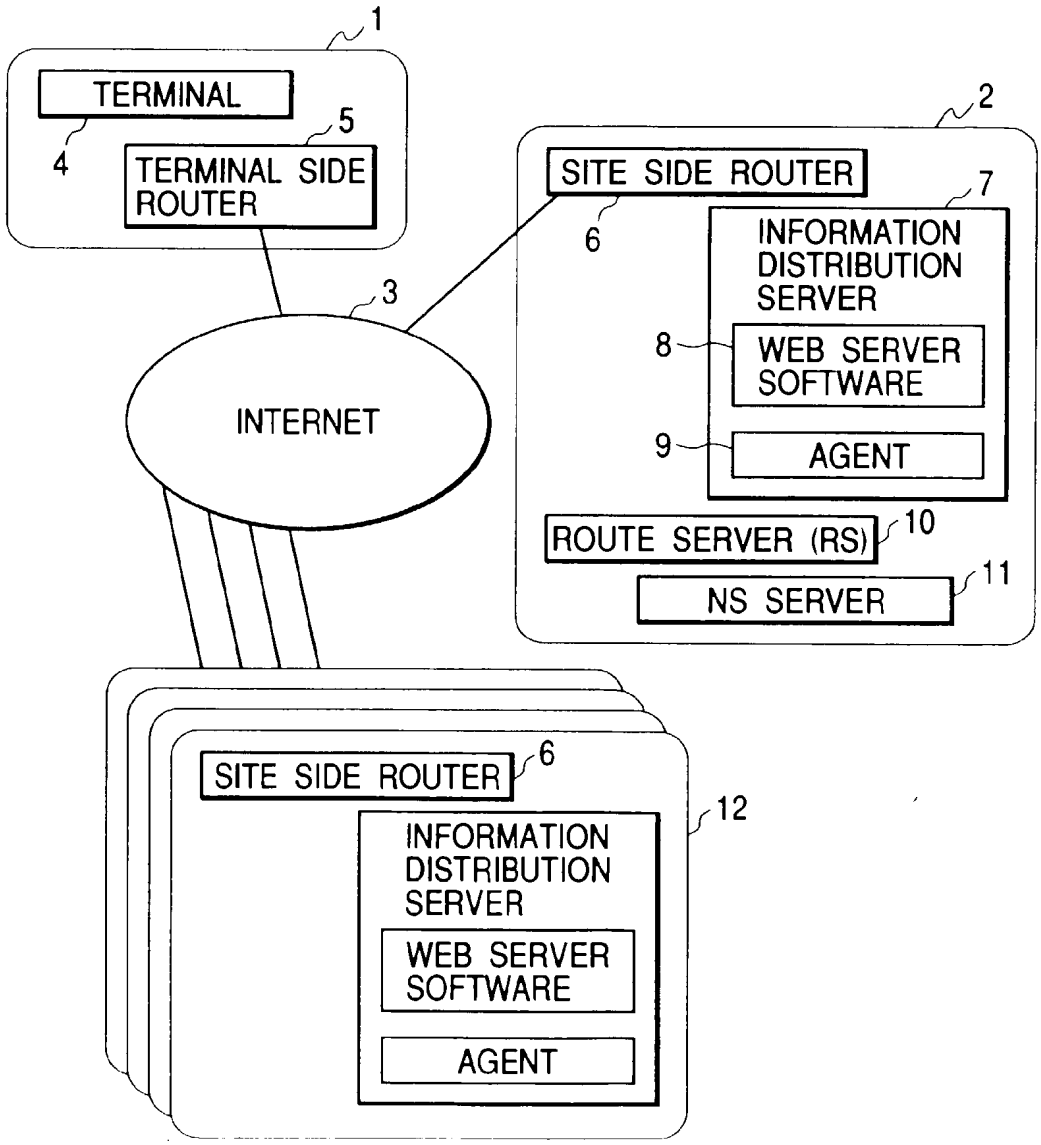

1
TERMINAL
4
5
TERMINAL SIDE ROUTER
2
SITE SIDE ROUTER
6
7
INFORMATION DISTRIBUTION SERVER
8
WEB SERVER SOFTWARE
9
AGENT
3
INTERNET
ROUTE SERVER (RS)
10
NS SERVER
11
SITE SIDE ROUTER
6
12
INFORMATION DISTRIBUTION SERVER
WEB SERVER SOFTWARE
AGENT

FIG. 2

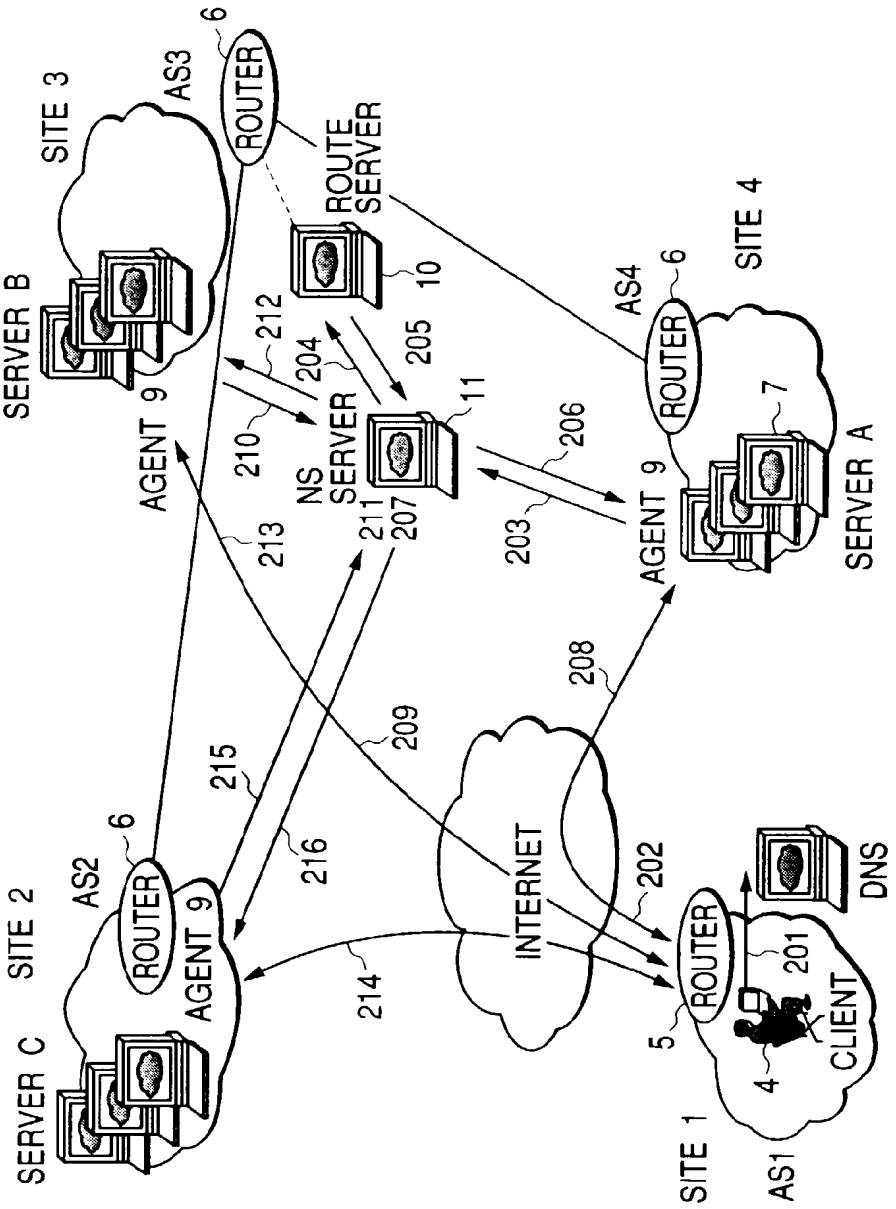

SITE 3
AS3
6
ROUTER
ROUTE SERVER
10
SERVER B
AGENT 9
212
204
205
210
NS SERVER
11
206
213
211
207
203
AS4
6
SITE 4
ROUTER
7
AGENT 9
SERVER A
208
209
215
216
INTERNET
202
SITE 2
AS2
6
ROUTER
AGENT 9
214
ROUTER
201
DNS
5
CLIENT
4
SERVER C
SITE 1
AS1

FIG. 3A

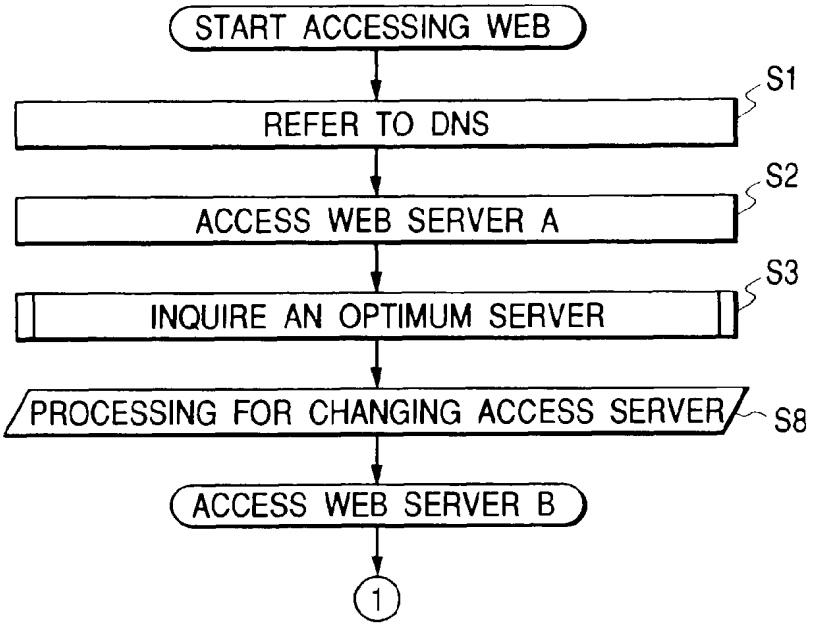

START ACCESSING WEB
REFER TO DNS
S1
ACCESS WEB SERVER A
S2
INQUIRE AN OPTIMUM SERVER
S3
PROCESSING FOR CHANGING ACCESS SERVER
S8
ACCESS WEB SERVER B
1

FIG. 3B

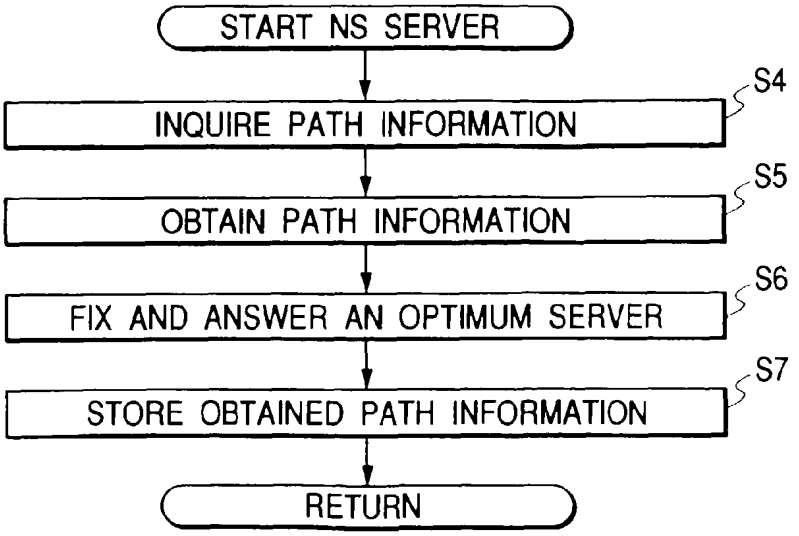

START NS SERVER
INQUIRE PATH INFORMATION
S4
OBTAIN PATH INFORMATION
S5
FIX AND ANSWER AN OPTIMUM SERVER
S6
STORE OBTAINED PATH INFORMATION
S7
RETURN

FIG. 3C

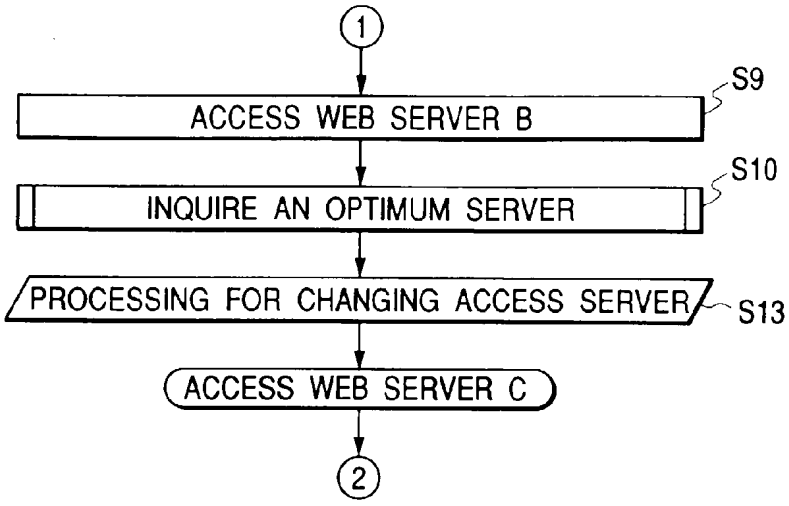


FIG. 3D

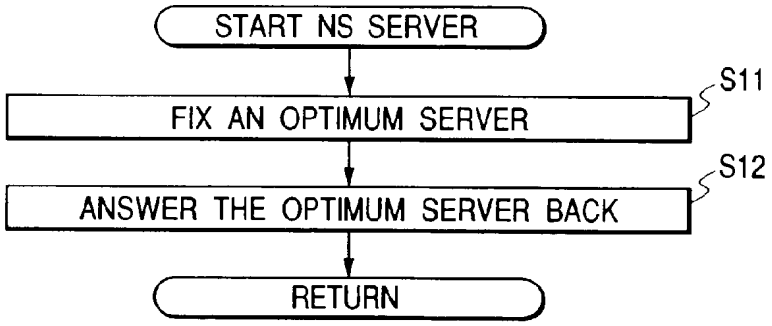


FIG. 3E

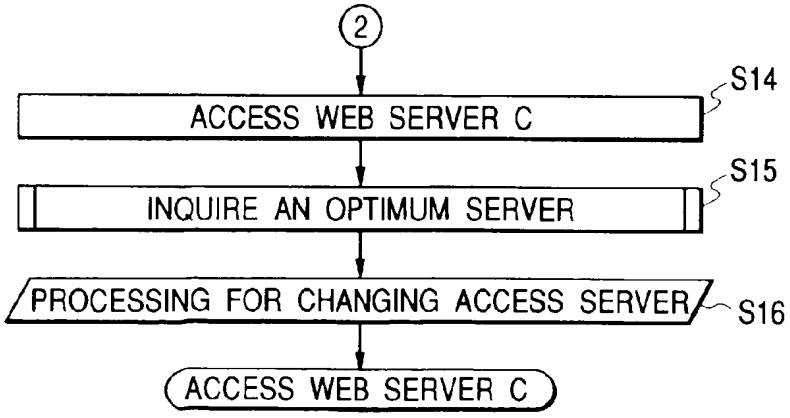


FIG. 4

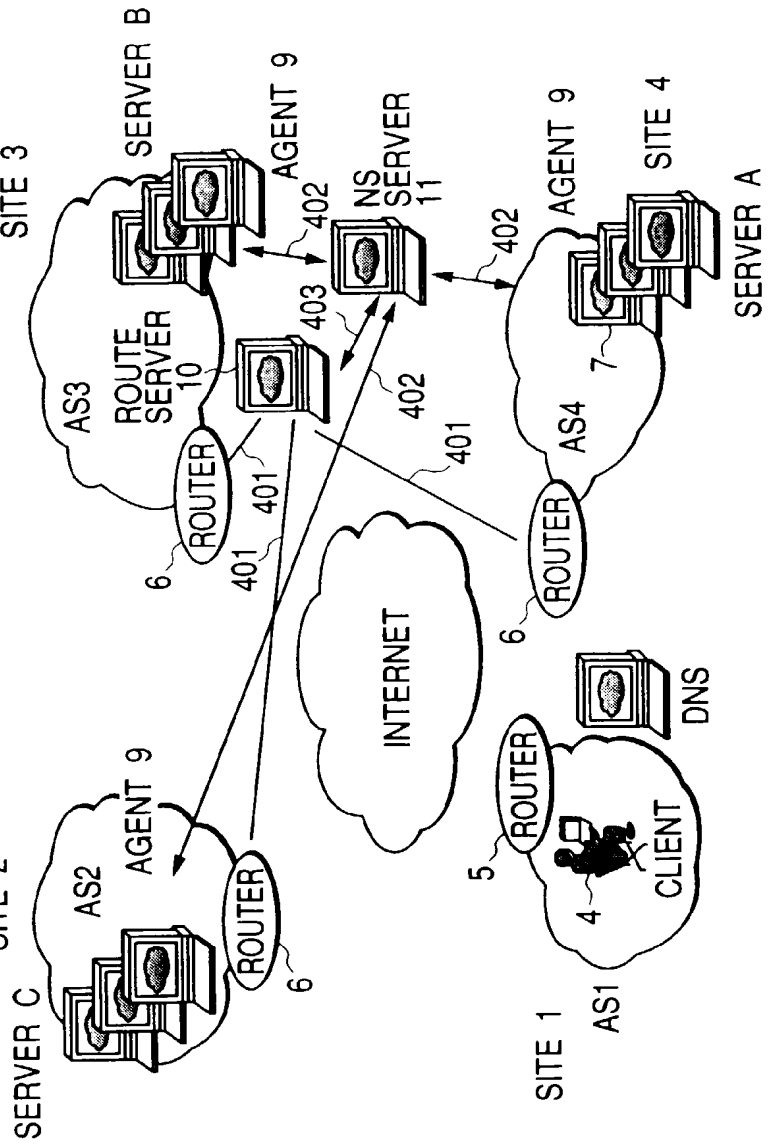

SITE 3
SERVER B
AGENT 9
NS SERVER 11
402
403
ROUTE SERVER 10
AS3
ROUTER
6
401
INTERNET
AGENT 9
SITE 4
SERVER A
AS4
7
SITE 2
AS2
AGENT 9
SERVER C
ROUTER
6
ROUTER
5
DNS
CLIENT
4
SITE 1
AS1

FIG. 5

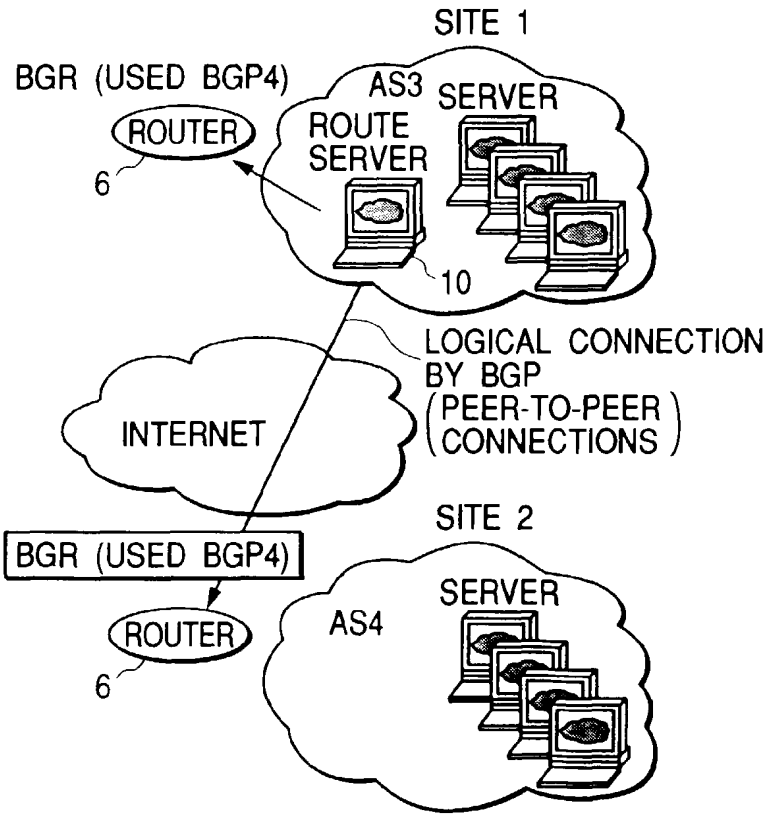

SITE 1
BGR (USED BGP4)
AS3
SERVER
ROUTER
ROUTE SERVER
6
10
LOGICAL CONNECTION BY BGP
(PEER-TO-PEER CONNECTIONS)
INTERNET
SITE 2
BGR (USED BGP4)
SERVER
AS4
ROUTER
6

FIG. 6

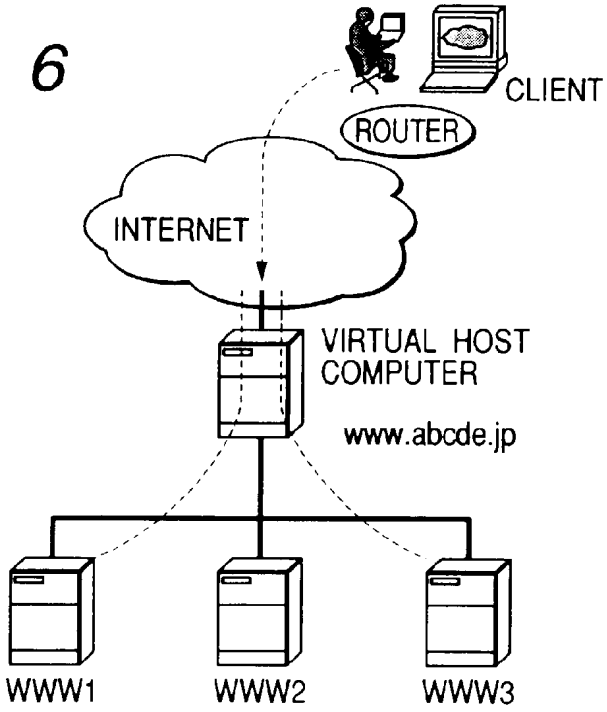

CLIENT
ROUTER
INTERNET
VIRTUAL HOST
COMPUTER
www.abcde.jp
WWW1
WWW2
WWW3

FIG. 7

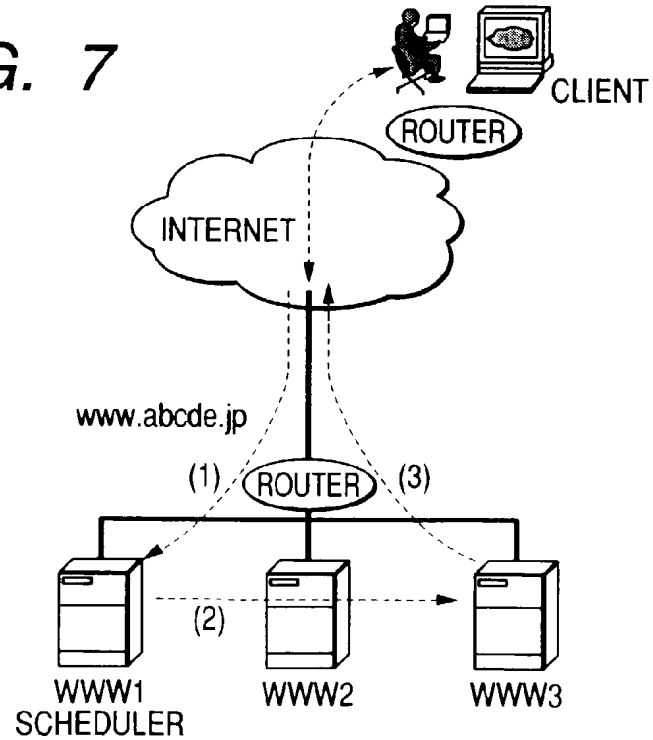

CLIENT
ROUTER
INTERNET
www.abcde.jp
(1)
ROUTER
(3)
(2)
WWW1
SCHEDULER
WWW2
WWW3

FIG. 8

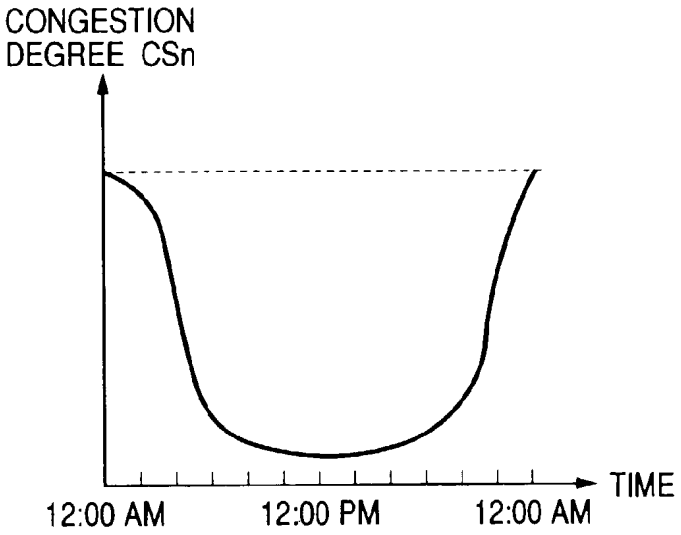

CONGESTION
DEGREE CSn
TIME
12:00 AM
12:00 PM
12:00 AM

FIG. 9

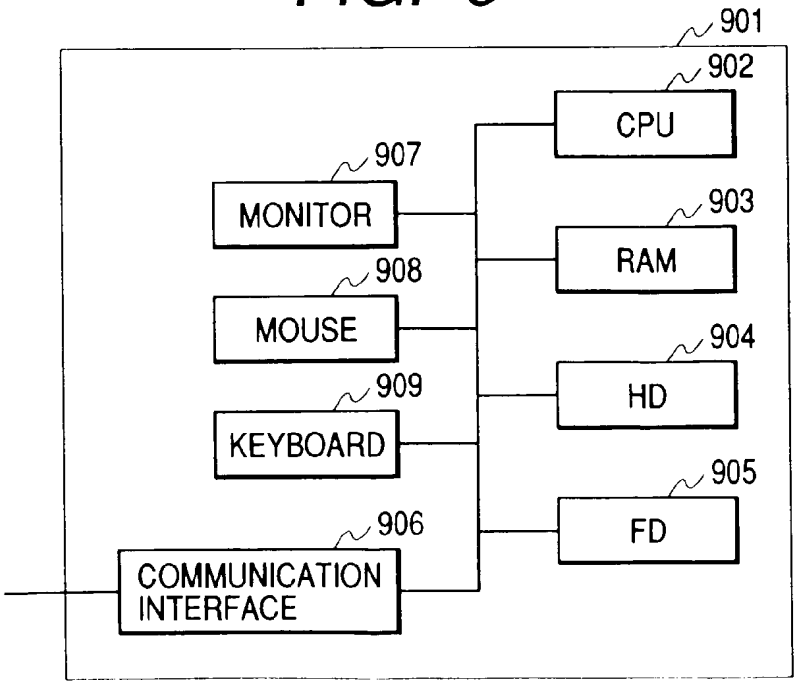

901
902
CPU
907
MONITOR
903
RAM
908
MOUSE
904
HD
909
KEYBOARD
905
FD
906
COMMUNICATION
INTERFACE

FIG. 10

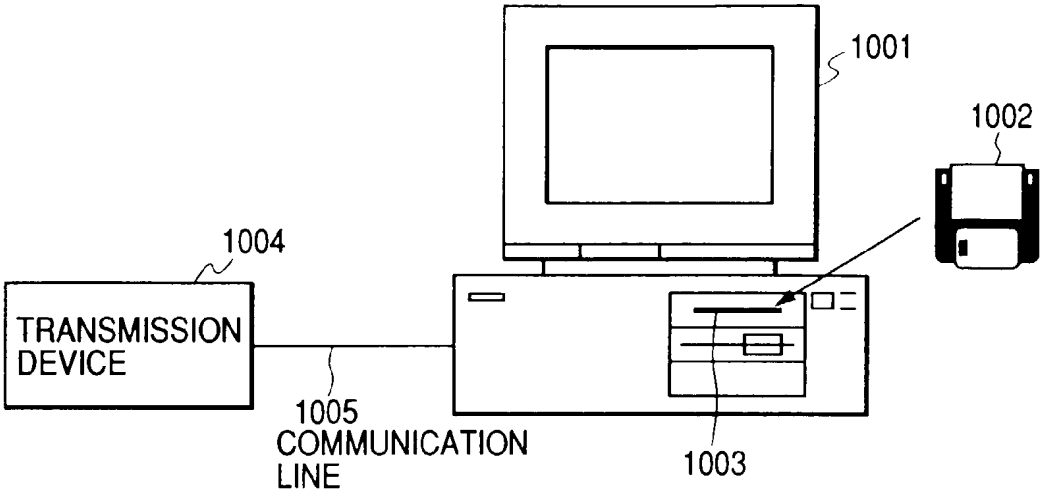
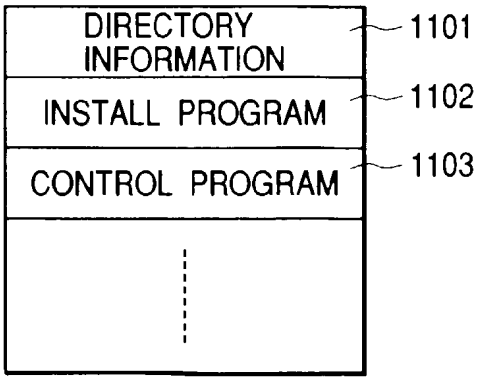

1001
1002
1004
TRANSMISSION DEVICE
1005
COMMUNICATION LINE
1003

FIG. 11


DIRECTORY INFORMATION
1101
INSTALL PROGRAM
1102
CONTROL PROGRAM
1103

[19]中华人民共和国国家知识产权局

[51]Int. Cl[7]
G06F 9/46
G06F 17/30

[12]发明专利申请公开说明书

[21]申请号 98807351.X

[43]公开日 2000年8月23日

[11]公开号 CN 1264476A

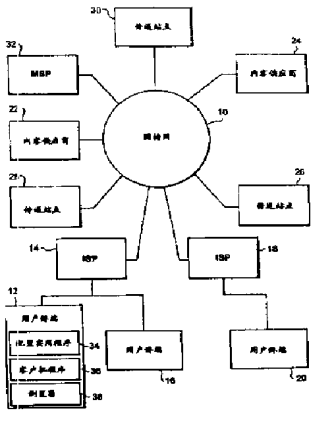
[22]申请日 1998.6.16 [21]申请号 98807351.X
[30]优先权
[32]1997.6.18 [33]US [31]08/878,385
[86]国际申请 PCT/US98/12784 1998.6.16
[87]国际公布 WO98/58315 英 1998.12.23
[85]进入国家阶段日期 2000.1.18
[71]申请人 英特维公司
地址 美国加利福尼亚
[72]发明人 布莱恩·肯纳尔 肯尼斯·W·科尔比
罗伯特·N·穆德里
[74]专利代理机构 中国国际贸易促进委员会专利商标事务所
代理人 鄢 迅

权利要求书 5 页 说明书 29 页 附图页数 4 页

[54]发明名称 分布式计算机网络上服务器端优化数据传送的系统和方法

[57]摘要

在分布的站点上，一用于优化视频数据存储和检索的系统和方法，要求遍布网 络配备“智能镜象”站点，每个镜象站点维护由系统管理的某些数据的拷贝。根据有关每个可得到的传送站点的网络性能分析，用户地址被赋到指定的传送 站点。一般的网络性能数据被收集和存储以便于选择另外的传送站点，确保同 传统网络相比保持有改进了的性能。



ISSN1008-4274

知识产权出版社出版

权　利　要　求　书

1.一种用于选择位于分布式计算机网络上的传送站点的系统，从传送站点接收一文件，该系统包括：

通信网络；

至少一个连到网络上的内容提供商，其中内容提供商存储至少一个文件；

至少两个连到网络上的传送站点，其中每个传送站点存储如同内容提供商相同的文件；

连到网络上的重定向服务器；

连到网络上的用户终端；

网络测试装置，用来测试网络和从连到网络上的所有传送站点中选择一优选的传送站点；以及

下载装置，用来从由重定向服务器指定的优选传送站点下载文件给用户终端。

2.根据权利要求 1 的系统，其中网络测试装置包括连到用户终端的第一存储介质，其中第一存储介质包含第一软件程序，用来测试网络及从连到网络上的所有传送站点中选择的优选传送站点。

3.根据权利要求 2 的系统，其中下载装置包括连到用户终端的第二存储介质，其中第二存储介质包含第二软件程序，能从优选的传送站点下载文件给用户终端。

4.根据权利要求 3 的系统，其中重定向服务器包括一数据库。

5.根据权利要求 4 的系统，其中第一软件程序使用由数据库建立的传送站点表。

6.根据权利要求 5 的系统，其中第一软件程序使用由网络分析过程产生的结果，以从传送站点表中选择优选的传送站点。

7.根据权利要求 5 的系统，其中第一软件程序使用由网络分析过程产生的结果，以建立一优选传送站点的优先表。

8.根据权利要求 6 的系统，其中网络分析过程至少部分在用户终端

上进行。

9.根据权利要求 8 的系统，其中网络分析过程包括很多网络测试。

10. 根据权利要求 9 的系统，其中网络测试从数据库中维护测试表中进行选择。

11. 根据权利要求 9 的系统，其中网络测试结果提供给数据库。

12. 根据权利要求 9 的系统，其中第一软件程序与用户终端结合实现网络测试。

13. 根据权利要求 11 的系统，其中结果被存储在数据库中。

14. 根据权利要求 13 的系统，其中数据库为用户终端映射一地址给至少一个传送站点。

15. 根据权利要求 14 的系统，其中第二软件程序与用户终端结合执行下载。

16. 根据权利要求 15 的系统，其中第二软件程序从数据库指定的服务器执行下载。

17. 根据权利要求 12 的系统，其中第一软件程序可在任何时候被执行。

18. 根据权利要求 17 的系统，其中第一软件程序可在任何文件被下载之前自动地执行。

19. 根据权利要求 1 的系统，其中文件包括视频剪辑。

20. 根据权利要求 7 的系统，其中第二软件程序使用优先表。

21. 根据权利要求 20 的系统，其中第二软件程序访问数据库。

22. 根据权利要求 21 的系统，其中第二软件程序首先试图使用优先表上指定的第一传送站点。

23. 根据权利要求 22 的系统，其中如果第一传送站点失败，第二软件程序将试图使用优先表上指定的后继传送站点。

24. 根据权利要求 23 的系统，其中如果优先表上所有的传送站点都失败，第二软件程序将试图直接从内容提供商下载。

25. 根据权利要求 1 的系统，其中文件具有与其相关的特征信息。

26. 根据权利要求 25 的系统，其中特征信息被存储在与文件对应的“EMBED”语句中。

27. 一种用于从网络上的大量传送站点中确定一优先传送站点集的方法，包括以下步骤:

获取一传送站点表;

执行至少一个网络测试;

处理网络测试的结果；及

根据测试的结果，把传送站点映射到 IP 地址。

28. 根据权利要求 27 的方法，其中执行步骤包括测试列在表中的某些传送站点。

29. 根据权利要求 28 的方法，其中执行步骤包括:

从表中选择传送站点的子集；及

为子集中的每个传送站点至少运行一次测试。

30. 根据权利要求 29 的方法，其中测试是从大量测试中选择的。

31. 根据权利要求 30 的方法，其中大量测试在连到网络上的数据库中被指定。

32. 根据权利要求 31 的方法，其中测试是根据数据库中规定的频率，从大量测试中通过随机方式选择的。

33. 根据权利要求 27 的方法，还包括把结果发送给连到网络上的数据库的步骤。

34. 根据权利要求 27 的方法，其中处理步骤包括对结果进行加权。

35. 根据权利要求 27 的方法，其中映射步骤包括对大量地址范围中的每个，按优先序列存储传送站点集。

36. 根据权利要求 35 的方法，其中每个地址范围对应于具有相同的第一字节的所有 IP 地址。

37. 根据权利要求 36 的方法，其中优先序列由测试结果确定。

38. 根据权利要求 27 的方法，其中传送站点表从数据库得到。

39. 一种用于从传送站点下载文件给用户终端的方法，包括以下

步骤:

标识所需的文件;

确定是否所需的文件在一个或多个传送站点上有镜象;

如果所需的文件有镜象，在映射到 IP 地址的传送站点表中寻找一优选的传送站点；及

从优选传送站点下载所需的文件。

40. 根据权利要求 39 的方法，其中确定步骤包括查询重定向服务器及接收一响应的子步骤。

41. 根据权利要求 40 的方法，其中查询步骤通过一脚本来实现。

42. 根据权利要求 41 的方法，其中脚本被嵌入到 Web 页中.

43. 根据权利要求 40 的方法，其中查询步骤由下载到用户终端的一程序来实现。

44. 根据权利要求 40 的方法，其中检索和下载步骤被重复直到下载步骤成功实现。

45. 根据权利要求 44 的方法，还包括如果重复检索和下载的步骤不成功则从内容提供商下载所需文件的步骤。

46. 根据权利要求 40 的方法，其中查找步骤包括以下子步骤:

为用户终端确定一地址;

将该地址与数据库中的地址表进行比较;

如果数据库中包含足够的数据，定位相应于该地址的优选传送站点；及

如果数据库中未包含足够的数据，定位一确省的传送站点。

47. 根据权利要求 46 的方法，其中比较步骤在主管该数据库的计算机上实现。

48. 根据权利要求 47 的方法，其中定位步骤在主管该数据库的计算机上实现。

49. 根据权利要求 46 的方法，其中第一定位步骤包括以下子步骤:

分离地址子表；并且

选择子表中的优选传送站点。

50. 根据权利要求 49 的方法，其中选择步骤在用户终端上实现。

说　　明　　书

分布式计算机网络上服务器端优化数据传送的系统和方法

本发明涉及分布式数据存储和检索的系统和方法。更具体的说，涉及一系统和方法，由此用户能获取一动态和分布式的多目标网络性能信息，而这种信息将被用来确认和选择最佳传送站点或服务器，从中接受计算机数据，特别是接受有关多媒体的内容。选用这样的传送站或服务器有利于增加网络容量，分散服务器端的负载，以及缩短服务器和用户之间传输的延迟。

因特网是将分散在全世界的计算机相连在一起的一个松散网络。通过指定目标地址，一条消息能从因特网上的任何一台计算机发送到另一台计算机，并经过一连串的“转发”，使消息在计算机之间传送。因特网上的每一台计算机、路由器或结点，都有一个唯一的因特网地址。当一台中间计算机或路由器接收到传输中的一条消息时，此计算机即检查这条消息将要前往的目标，并相应地将它往前传递。

因特网不仅在规模，而且在其复杂性方面都在急速地发展着。过去，因特网的大多数用户为学者、研究人员或公共团体用户。当时的因特网主要用作传送和接收电子邮件、网络新闻和允许传输计算机文件。然而，自从几年前万维网（也被称作“Web”或“WWW”）的兴起，因特网上开始引入能引起大众兴趣的其他数据类型，即图像显示，商品信息等等。

Web 协议和 Web 语言建立了一种以图形作为航行于因特网浩翰的海洋的手段。通常主要是由文字和图形材料组成的“Web 页” 被存储在无数遍及因特网的称之为“Web 服务器”的计算机上。通过指定想要查看的 Web 页的地点（即因特网地址），一种被称为“浏览器”的软件可以被用来访问和查看因特网的 Web 页。当 Web 页被访问时，Web 页上的信息从远端的计算机（服务器或传送站），不管这台计算机

在世界上何处，通过因特网传送到给用户。

近来，许多非常复杂类型的多媒体内容开始出现在网上，例如音频和视频数据以及计算机软件。与第一代 Web 内容（即文本和静态图像）相比较，音频剪辑、视频剪辑以及软件程序对存储量和带宽有极高的要求。

目前，在因特网的多结点链接上要支持大容量音频/视频文件的高速传输是困难的。因为数据经常是从遥远的地方传过来，其间有许多因素能够造成延迟或者甚至于部分或全部数据的丢失。一般情况下，用户在接受小图形或文本文件时，如果感到有些微小的延迟是不危险的。然而，显然象视频这样的实时数据对数据传输和显示有着非常明确而严格的定时要求。

不幸的是当前设计的类似于传统因特网的数据网络基于此这样的原则，即延迟和重大的数据传输率的变化对于普通数据（如文本和静态图像）来说是可接受的。因此，由于允许从世界上任何地方获取文本和图形信息有着重要价值，这些传输的缺陷被认为是可以接受的。为了降低数据传输费用，因特网的基本容量有一些“超额”。换句话说，为了开出相对低微的远距离通讯连接的总费用，网络数据传输的及时性明显地已被折衷了。

为了成功地通过象因特网这样的面向消息的网络来传送音像数据，对于任何多于几个的用户，网络资源应能以便于及时传输的方式供使用。一个利用投入的网络资源的系统通常不能利用现存的如因特网共享网络的定价方案，由于它不能参与以逐个数据包为基础的网络资源共享。视频数据必须被排除在更低的优先权的数据之外而被传输。因而传输费用变得重要，尤其是当“远距离”连接时，或者是连接持续了很长一段时间时。

以上讨论的及时性对费用折衷的另一个后果是看来杂乱的网络拓扑设计。由于延迟和吞吐量不稳定总是在传统上被归咎于偏爱低成本，因特网基础设施的配置同样受成本考虑的驱动。相应地，网络的互连效率很少被考虑。迅速增长的实时数据正在改变这种需求。

应该承认，在因特网上时间敏感性数据的不充分的数据传输性能，通常由以下四种原因引起：数据包丢失，过度的服务器利用，相对低容量的网络基础设施， 以及网络硬件中的内在延迟。数据包丢失，更具体地讲，是由基础设施的不足和路由缺乏鲁棒性而造成的。内在延迟则被认为除了别的以外还有因特网上多结点路径中邻近结点间缺乏流控而造成的。

不同于较小的文本和图形文件，相对较大的视频文件将需要几分钟（甚至更多）的“流”，或恒定数据流。所以，通常网络性能问题恶化了。网络带宽或特定网络的数据运载容量受到限制。因此，加剧了数据包丢失和数据延迟。长的传送时间将长时间地消耗大量的服务器容量，减少其它用户的可用资源。从而，由于网络的基础设施变得越来越拥挤，数据包的丢失与延迟继续加剧，相应地传输时间增加了，服务器的负载也更进一步加重。

这种模式例证了网络性能的“螺旋式向下”，这种性质受着企图传输如视频剪辑这样的大文件的驱动。只要网络通信流量仍然还保持在网络带宽的限制范围内，网络性能将仍然可接受的。然而，一旦网络负载的高峰超过其容量，上面所述的“螺旋式向下”将会发生，造成低网络性能期的延长。

如前面已讨论的，指定所要的 Web 页的位置（即因特网地址），或者用一种更普通的说法，通过到 Web 页的“热链接”，一个浏览器程序可被用来通过因特网访问和查看 Web 页。一般地浏览器有 Lynx，NCSA Mosaic，Netscape Navigator 及 Microsoft Internet Explorer。所要的 Web 页由一个叫做统一资源定位器（“URL“）的来指定，使用的句法“http://internet.address/directory/filename.html”指示文件的精确位置。

Web 页通常用一种叫做“HTML”（超文本标记语言）的语言表达出其布局和内容。任何一台连到因特网上特定的计算机可存储一个到多个 Web 页，也就是一些按 HTML 格式的计算机文件，以供用户访问。

从一个 HTML Web 页到另一个 Web 页的热链接是按如下的步骤来完成的，用户首先访问一个已知地址的 Web 页，通常是放在用户的 ISP（因特网服务提供商）那里的一台计算机上，ISP 即是为用户提供因特网连接的组织。除了用 HTML 格式规定的文本的和可视的数据，Web 页还能包含“链接”，即嵌入指向其他 Web 页的因特网地址的信息（以 URL 的形式），这种 Web 页经常在遍布因特网的其他计算机上。用户通过选择一个链接（通常用鼠标点击），就能访问其它 Web 页，在这些 Web 中依次又包含更多的数据和/或其它的链接。

对 HTML 的各种各样的扩展形式，如 Netscape 的 EMBED 标记，允许访问嵌入到 Web 页中的其他数据。某些浏览器不能处理除文本和图像数据以外的数据，而有些浏览器却能处理各式各样的数据。例如，NCSA Mosaic 处理未知类型的数据的方式是，将数据下载到用户的计算机上，然后有选择地调用一个外部程序来查看或处理这些数据。最近发布的 Netscape Navigator 与 Microsoft Internet Explorer 将这种观念更往前发展了一步：一个浏览器扩展，或称为“插件”，当接收到远端 Web 页的数据能够自动地被调用以处理此数据。还有其他的方式，如用 Java 语言(或其他类似语言)书写的网络程序“applets”，能用来延伸浏览器环境功能或网络的功能。

数字多媒体数据对存储量和带宽有极高的要求，尤其是视频文件可能会很大，大约从 10M 到 10G 字节。为了在用户的终端以接近其记录速率的速度来播放视频文件，必须保证传输文件的高速与恒定。如果速度太慢，图像播放起来将会慢于原始记录；如果速度不稳定，视频会看起来不平稳，正如旧时的电影一样。

以上讨论的网络设计的折衷通常反过来影响在因特网上传输音像数据。当一个用户使用浏览器在网上“冲浪”时，不会注意到接受文本和静态图像中较小的延迟和传输速率的变动，而这样的缺陷在获取实时音像信息时就变得明显和至关重要了。

为了解决这种问题，因特网内容提供商们有时在不同的服务器和传送站点（被称为“镜像站点”）上面，以因特网为基础传播受欢迎的

内容。每一个镜像站点实质上包含了与源站点完全相同的信息。例如，如果一个流行 Web 站点的位置在纽约，其镜像站点可能在洛杉矶，伦敦和东京。相应地，如果一位欧洲用户访问在纽约的源站点有困难时，他能通过热链接到达地理位置最近的镜像站点，也就是伦敦。

然而，镜像站点有一些不利的因素。例如，镜像站点也许有着广阔的地理分布，但并非以实际用途和网络通信量等因素出发在因特网上高效分布。因而，纽约和洛杉矶的镜像站点也许是连接在同一国际因特网服务提供商的网络上的，这就意味着若无法访问其中的一个站点，可能影响访问其它站点。

此外，为了减轻每个服务器的负载，镜像站点可能并非放在最佳地点。尽管，对于一个镜像站点应该放在何处事先已经做过“有根据的推测”，但从实际用途的角度来看也许会有偏差。并且不保证增强网络性能。镜像站点的带宽可能会不如原始站点，它可能由于其他的原因而超负荷。

还有一点，镜像站点通常被随意作主。如果一个 Web 站点非常受欢迎，则网络服务提供商认定为其主题材资料也许会增加订购者的兴趣，服务提供商就会同意对原始 Web 站点的镜像站点作主。这样的安排将吸引对镜像站点作主，因为人们将被吸引到镜像站点上，同时也就可能热链接到作主的其它内容上。从另一方面来讲，这样的随意联合通常并不可靠，不能在任何时候都提供服务。

实质上，镜像站点提供数据的二级资源，也许可利用，也许不可用，这可增加用户的方便，但不能处理网络带宽和效率。若仍然采用如因特网上目前存在的低成本价格的配置，一个镜像站点是说明不了网络性能特征的，同样也不等同于可用于高效地传输视频数据的有效带宽。

当前，没有一个如何选择传送站点最佳位置的指南，也没有一个已知的方法可以让用户决定，究竟连接到哪一个镜像站点才能够保证最优性能。实际上，采用传统的镜像站点是一种随意的方式。通常地，一个用户会尝试着访问一个原始站点（或一个已知镜像站点），只有当

经过一次或数次尝试后，发现这样的连接性能低时，才会转向另一个镜像站点。这种方法是对网络资源的低效利用。毫无疑问，镜像站点并非解决 Web 站超载问题的最佳方案。其中一个首要原因就是缺乏对网络性能的考虑。

网络分析，尤其是对因特网上确定路径和链接的性能的分析，是一种知名而被开发的方法。例如，一个叫做“Ping”（强制回应）的程序可以让连接到因特网上的计算机来确定一个远端主机是否可访问。然而，“Ping”（强制回应）程序使用了一种名为 ICMP 协议的低优先级的网络协议，同时它并未相应地提供给用户有意义的性能分析信息。另一个叫做“traceroute”（跟踪路由）的程序追随从一台计算机到远端主机消息的传输，跟踪沿着每一个链接的延迟，从而确定这条消息所取的路径。Traceroute（跟踪路由）应用能够被用来映射数据流。然而，它缺乏提供有意义的网络性能分析信息的能力。Traceroute（跟踪路由）只提供了消息的单方向传播的路由信息，而且只是传播瞬时的信息。

此外，典型的判断只在单机上反复进行路径连接特性的运行测试。然而要将测试范围扩展到整个因特网虽然有可能，在逻辑上是不可实现的，因为因特网的范围如此之大。

传统的网络分析技术如“Ping”（强制回应）程序和“traceroute”（跟踪路由）程序，都从网络连接的角度出发，但几乎没有提供从因特网上的服务商和镜像站点得到什么样的性能信息。因此，只有进行“推测”来确定传送站和镜像站点应该安放在什么位置，或应该使用哪一个镜像站点来优化网络性能。

因此，需要找出一种方法来测定整个网络性能。还需要找出一个系统来，应用这种方法使内容提供商可以动态地确定传送站或镜像站点所在最佳网络位置，同时允许用户选择最佳镜像站点，从它来接收数据。

本发明指出了一系统和方法，以根据因特网定位站点，优化 Web 内容的分布。一个智能的镜像方案，在这里称之为“智能镜像”，可用

于测定对镜像站点的需求和分布，并且指引用户到最佳的镜像站点请求某些 Web 内容。

许多“智能”的传送站点和镜像站点被用来分配受欢迎的 Web 内容到因特网上的不同部分。经过大量的用户测试过的全面的网络分析方案，被用来交互式地确定站点的最佳位置，以及确定每个单个用户应使用的最佳站点。

相应地，由于每个单个用户被路由到提供改进性能的智能镜像站点或传送站点，整个网络的堵塞也就减轻了。在大多数情况下，改进的服务器位于电学上离用户位置比较近的地方，以便于减少数据必须经过的网络连接的数量，从而减少了数据包的丢失和延迟。

此外，网络分析结果将允许消息通信量从那些已经过载的传送站点和网络区域引导开，转向未充分使用的服务器和网络区。这样的结果从每个用户来看改进了吞吐量，从而增加了利用本系统的内容提供商提供内容的吸引力。 内容提供商们能触及因特网上许多的用户，并且性能不遭受显著地降低。

根据本发明的系统从一个原始站点和至少一个另外的传送站（或镜像站点）开始。每个要使用本系统的用户都将被提供使用，在优选实施例中，使用一个包含配置实用程序和客户机程序的软件，配置实用程序首先用于确定哪一个传送站点为具体的用户提供改进的性能。

在本发明的实施例中，配置实用程序首先从服务提供商处下载一个“传送站点文件”。这个传送站点文件包含了一个可用的传送站点列表与一个将运行的网络测试列表。进行测试的类型与测试的频度可在这个传送站点文件中指定，诸如，取决于测试网络的用户数和估测的网络流量或传送系统的容量。

这个配置实用程序将运行传送站点文件中指定的测试子集。测试结果可以显示出哪一个传送站点对用户有更好的性能，并且也包含了从运行此测试的用户角度出发的各种一般化的网络性能信息。这个网络测试结果和所选择的传送站点的身分将被发送回传送服务提供商（通过 Email（电子邮件）以一个可能实现的配置），合并进入服务提供商

的数据库中。

由配置实用程序选出的传送站点则被用户采用，以重新得到那些被传送系统提供商管理的内容。因此，当用户浏览 Web 页内容与查找一个特定的条目的时候，如视频剪辑这样被服务提供商的传送系统管理的内容，客户机软件将自动从指定的“智能镜像”传送站点检索它。站点的优先权和缺省的站点可以周期性地在指定时间根据请求，或响应网络负载与流量的变化而更新。

此外，因为本项发明的配置实用程序执行各种各样的网络测试，并将测试结果提供给服务提供商，关于系统和网络性能的有价值的数据是可以得到的。这些数据提供有关哪些智能镜像传送站点是有效的，哪些是无效的，哪些智能镜像传送站点超载，和因特网中什么部分可从增建更多站点或容量而得到好处。这些数据也可能去完成那些复杂的网络分析，比如端到端的性能估测，工作负载特性，路由的稳定性和中断的度量。

在本项发明的实施例中，镜像服务提供商利用终端用户提供的网络性能数据，将因特网 IP 地址与“电学上接近”的传送站点相关连起来，获得一张查找表。当用户正在浏览 Web 页，并请求一个文件时，例如被服务提供商的传送系统管理的一个广告标题或视频剪辑，服务提供商可以映射该用户的 IP 地址到查找表，并决定哪一个传送站点和用户在电学上是接近的，服务提供商则可为用户的配置实用程序或客户机程序提供这些服务器的单个传送站点地址或传送站点地址表。对后者，用户终端起路由器作用作出最终的传送站点选择。

一般地，为特定的用户改进的传送站点，通过分析聚集一组由用户预先完成的网络测试中收集的网络性能数据，能提前被预测。因而，每次用户请求一个被镜像服务提供商的传送系统管理的文件时，传送站点的选择可能在进行过程中发生。从用户的角度看，传送站点的选择自动地透明地进行，以致看起来从 Web 页上选择一文件和使文件传送到用户终端之间没有延迟。由服务提供商维护的查找表不断地更新，以反映网络性能变化，使得服务提供商有可能有效地平衡网络流量负

载。

因此，从工程的角度看，镜像服务提供商可以持续地保证提供经过改良后的性能。从市场角度看，还可以告知内容提供商，为了改进性能何处去定位镜像站点或传送站点，以及哪些 ISP 能提供改进的传送。

图 1 是根据本发明描述的系统的网络拓扑方框图，它包括有多个用户及多个内容提供商；

图 2 是一个流程图，描述了根据本发明的一系统中所用的配置实用程序的操作；

图 3 是一个流程图，描述了根据本发明的一系统中所用的客户机程序的操作；及

图 4 是一个流程图，描述了在本发明的一实施例中如何实现站点的选择。

参考详细描述的实施例，本发明被说明如下。显然，本发明可以各种各样的形式而体现，其中某些可以完全不同于公开的实施例中的那些。因此，在此公开的专门结构和功能细节仅是一代表性的，并决不限制本发明的范围。

首先参考图 1，因特网 10 通常代表广域通信网，被描述为“云(cloud)”。因特网为大家所知是大量计算机的一互连网络。尽管“地理上”彼此接近的因特网互连计算机可以在因特网上“电学上”彼此接近，但这不是通常的情况。而且，连到因特网上的一台计算机能同连到因特网上的任何其它计算机通信；在彼此直接互连的计算机之间，消息极像在包含有一系列链或“路程段(hops)”的通路上传送。

第一个用户终端 12 在图 1 中也被描述。第一个用户终端 12 被连到因特网服务提供商（ISP）14，它通常仅仅是一台计算机、路由器、或是连到因特网 10 的终端服务器。ISP 14 能主管额外的用户终端，例如第二用户终端 16。其它的 ISP，例如第二 ISP 18 也连到因特网 10 上。第三用户终端 20 被显示连到第二 ISP 18。这里仅显示了三个用户终端，然而，应该知道，依据下面阐述的操作细节，本发明同时存在

的用户数是不受限制的。

熟习专业的人都知道，内容提供商也被连到因特网 10。第一内容提供商 22 可提供某些类的内容，如运动得分及精彩场景。第二内容提供商 24 可提供不同类的内容，如商务新闻。

传统上，如果一用户（如使用第一用户终端 12 者）想要访问第一内容提供商 22 提供的内容，终端 12 可直接地查询第一内容提供商 22。请求消息将从终端 12 通过因特网 10 传给内容提供商 22，内容提供商 22 将通过因特网 10 把所要的数据送回到终端 12。

几个传送或“镜像”站点被显示连接到图 1 的因特网 10 上。第一传送站点 26 可能处于距第一用户终端 12 少量“路程段”的位置。第二传送站点 28 可能位于远离第一个用户终端 12，但接近第三用户终端 20。第三传送站点 30 也许如第二传送站点 28 那样接近第三用户终端 20。如前所说，用户及提供商或传送站点在因特网上“地理上”彼此接近，但在“电学上”彼此可能并非接近。通过减小用户及提供商或传送站点之间“电学的”距离，数据必需在上通过的网络连接器及路由器的数量能够减少。

如上所讨论的，通过减少前述的网络问题的发生率，智能镜像系统起改进网络性能的作用。尽管在某些情况下，本发明的网络测试过程显示某些较长的通路比某些较短的通路提供了更好的吞吐量，通过减少数据必需在上通过的网络连接器的数量，包丢失及延迟问题通常也被减少。出现在网络电缆中的包丢失非常少及基本上没有延迟，它们典型地由超负荷的网络存储器及路由设备引起。由于本发明的智能镜像站点被定位成电学上彼此接近每个用户，减少了包的丢失及延迟。因为多个传送站点共享通常已被单个服务器操作的负载，过量服务器利用问题得以减少。相对低容量的网络基础设施变得不是大问题，因为从不同位置的并行传送站点检索的数据通常不再需要在相同的网络链上传送。

为了描述本发明，传送站点是网络上的一“结点”，它可存储数据和其它文件，如软件码，用于传送。该项也可包括负责数据传送的站

点，包括镜像站点，内容提供商，以及广播视频流服务器或 Web 站点。

在系统中，镜像服务提供商（MSP）32 被连到因特网 10。MSP 32 包括能在因特网 10 上发送和接收数据的一数据库，在传送站点 26，28，及 30 的分配上，和从用户终端 12，16 和 20 上请求对原始端及传送站点定位上行使管理功能。

通过使用在用户终端 12 上的存储介质中（如随机存取存储器）运行的配置实用程序 34 和客户机程序 36，这种管理功能变得更为方便。尽管图 1 所示的配置实用程序 34 和客户机程序 36 仅是第一用户终端 12 的一部分，应该承认，任何参于此系统的用户终端，如用户终端 16，和 20 将使用这些软件。决定参于系统的用户能直接从 MSP 32，或通过传统零售的或其它渠道（如浏览器或计算机操作系统）得到构成实用程序 34 和客户机程序 36 的软件。应该说明，在本发明所述的实施例中由配置实用程序 34 实现的功能能集成到通常的因特网应用软件中，如浏览器或其它网络应用中，独立的程序是不必要的。

在优选实施例中，在用户终端 12 访问系统之前，配置实用程序 34 必需通过用户，或是通过命令，或者是自动地运行。配置实用程序 34 的操作详示于图 2 中。

当配置实用程序 34 首次在用户终端 12 上运行时，从 MSP32（图 1）检索传送站点文件（步 40）。如果用户已经有了传送站点文件（例如曾用配置实用程序 34 检索出），而且传送站点文件足够新，则传送站点文件可从用户终端 12 的当地硬盘中检索。传送站点文件包含所有可用的传送站点表（如传送站点 26，28，及 30）及要在用户终端 12 上运行的网络测试表。在本发明的范围中，可以少到只有两个传送站点，或按用户数进行调正，而多到有几千。原理上站点的数量是不受限制的，每个可用的传送站点被表示在传送站点文件中。 传送站点文件从 MSP 的计算机系统中通过数据库生成。数据库应用使用有关用户的信息，动态地确定运行的最佳测试。因此，传送站点文件不需要包含现有的每个传送站点入口，该表能简化到只包括显现合适或切实可行的的那些站点。

开始，对传送系统的用户，测试配置中运行时间变化的大小是小的；也就是说第一组用户全部运行基本上是同样的测试。而且，随着传送服务增长，每个用户的测试强度减少，以补偿测试网络宽度增加的幅度。测试的范围及被测的传送站点的数量两者都可能变小，以进一步减少网络测试的总负担。

在一个实施例中，传送站点文件通常具有的格式如下：

1. 文件的修订次数及消息。文件包括这种字段以判断是否配置实用程序 34 的新版本是合用的。如果在传送站点文件中修订次数高于配置实用程序的修订次数，配置不被允许，代之的是用户将被提示去获取一更新版本的配置实用程序 34。在此描述的文件版本的确认，确保最新的传送站点选择算法被用于由配置实用程序 34 生成的测试数据中。

2. 一可用的智能镜像传送站点表，对每个可用的传送站点，提供下面的信息：

a. 主机名。按已知的 www.server.com 的因特网格式。

b. IP 地址。按已知的数字因特网地址格式。这地址目前是 32 位数的 w.x.y.z 的形式，这里的 w,x,y,和 z 均为 0 到 255 范围内的数。

c. 替代名。一个不正式名如“第一镜像站点”。

d. 要被执行的测试表。对每种测试提供下面的信息：

i) 测试 ID。每种测试类型具有为配置实用程序 34 所知的唯一标识符。

ii) 权重因子。每个测试将用指定的百分数加权。

iii) 频率。每个测试并不需要每次运行。该字段指出一概率，确定一特定的测试运行的频度。

iv) 附加信息（选项）。对某些测试，附加信息可能需要。

e. 站点优先级别。每个站点可给定一权重或优先权级别，如在 1 到 100 之间。如下面所讨论的，在 MSP 的数据库中聚集数据用于执行网络使用分析，不可能只用单个用户的瞬间端对端测试。这里提供的权重因子被用于合并从服务提供商的数据库接收的测试结果。一旦预

定的最大用户使用级已经达到，该权重因子也被用于限制赋新用户给传送站点。

f. 测试站点标记。如果该标记为允许，前述的测试将被运行，但该站点将不会被赋予作为传送站点，即使它产生最佳的性能。

g. 内容提供商组。每个站点能属于一个或多个内容提供商组，因此，只镜象某些内容。如果一个用户并不关心由特定的站点作主管的数据类型，那末它就不需要被测试。

然后，配置实用程序 34 询问用户（步 42）在配置过程中所需的各种信息项，如用户名，电子邮件地址，口令，调制解调器的速度，及同访问控制相关的信息（如各种属性中的那些级能被用户看到）。访问控制机制将在下面作更详细的讨论。在本发明的一实施例中，从用户接收的信息被加密并存储在用户终端 12 的配置文件中。

然后，配置实用程序 34 判断是否用户终端 12 被连到因特网（步 42）。如果否，它将启动连接（步 44）或提示用户去作。

然后，执行一系列的网络测试（步 46）。对传送站点文件中列出的每个可用的站点，可执行一个或多个测试；并非所有在文件中的站点都需要被测试。

目前认为下述测试类型提供有用的数据:

1. Ping（强制回应）. 提供关于远程服务器是否是可得到的信息，如果是，则对低优先级的消息从用户终端 12 传到远程服务器及反回的往返传送它要化费多长时间。Ping（强制回应）是一简单测试，用来确定一站点是否可得到进一步的评估是可用的。由 Ping（强制回应）应用返回的过多的时间能被用来淘汰对有效信息传送太“慢”的传送系统。由终端使用的这种测试减少了被测试的传送站点数。

2. Traceroute（跟踪路由）. 提供一消息，从用户终端 12 到远程服务器取什么路由的信息，包括沿此通路什么系统被使用，及每个路程有多长。Traceroute（跟踪路由）被配置实用程序 34 使用以给出信息传送路经的资料。几个具有不同结果的跟踪可以指示从一特定用户到指定的服务器的路经的稳定性是不可接受的。前面来自服务提供

商的系统数据库的特定路径上的聚集数据，也可影响为具体用户选择特定传送站点的确定。路由的稳定性是首先要考虑的。

3. Reverse Traceroute（反向跟踪路由）.提供一消息，从用户终端 12 到远程服务器取什么路由的信息，包括沿此通路什么系统被使用，及每个路程有多长。Reverse Traceroute（反向跟踪路由）被配置实用程序 34 使用以给出信息接收路经的资料。几个具有不同结果的跟踪可以指示从一特定服务器到指定的用户的路经的稳定性是不可接受的。前面来自服务提供商的系统数据库的特定路径上的聚集数据，也可影响为具体用户选择特定传送站点的确定。再次说明，路由的稳定性是首先要考虑的。

4. Dynamic Traceroute（动态跟踪路由）. 类似于跟踪路由或反向跟踪路由，但只是在任何因特网上指定的计算机对之间。动态跟踪路由被配置程序使用以给出信息传送路经的资料。几个具有不同结果的跟踪可以指示两个网络位置之间路由的稳定性是不可接受的。前面来自服务提供商的系统数据库的特定路径上的聚集数据，也可影响为具体用户选择特定传送站点的确定。如上所说，路由的稳定性是主要要考虑的。

5. 名字服务器解析延迟。如果未指定数字因特网地址，名字服务器执行查找以确定什么数字地址对应所要的主机名。此处理过程能化费大量的时间。

6. 吞吐量。一样例文件从远程服务器被下载或部分下载，以确定以每秒字节数为单位的实际吞吐量。

7. 吞吐量的变化。一样例文件从远程服务器被下载或部分下载，以确定吞吐量是相对固定还是上下变动的。

8. 出错率。一样例文件从远程服务器被下载或部分下载，以确定传输是否遇到传送错误。该信息通过对返回到用户的错误信息包计数而得到，从而得出从用户到服务器的出错率，以及通过跟踪接收的传输字节碎片的字节数，从而得出从服务器到用户的出错率。

9. 包破碎。一样例文件从远程服务器被下载或部分下载，以确

定是否传输遇到破碎或接收到乱序包。

10. 容量查询。如果远程服务器这样被允许，配置实用程序 34 查询服务器以确定它的传输容量及它的平均负载。该信息是通过近乎所有网络服务器都支持的简单网络管理协议（“SNMP”）收集的。

11. 细微的网络分析。由 MSP 数据库累加的数据提供一全局观点的网络行为。该信息允许智能镜象系统用户以历史的观点观查可用的传送站点。累加的数据由传送系统的数据库管理的，以分析网络的性能，为的是强化在网络高容量区域的使用，而不强调在已经历性能减少的区域中的使用。

关于如何实现前面的每个测试的信息在网络分析技术中是熟知的。如可见 Bob Quinn & Dave Shute, “Windows Sockets Network Programming” (Addison – Wesley 1996)。在本发明的一实施例中，测试是通过执行“Ping”（强制回应） 测试以检验是否一服务器可获得而完成。一系列小的（如 20k）下载，一系列大的（如 200k）下载，以及“Traceroute”（跟踪路由）和“Reverse Traceroute”（反向跟踪路由）测试，以得到传送路径的资料。

跟踪路由信息被 MSP 32 使用，把测试数据与它的数据库中的信息相互连系，这样，特别差的网络链及服务器能被识别。这些信息提供在上面讨论的传送站点文件中。根据从其它用户处得到的信息，如果知道一特定的链或服务器不可靠，即便单个测试给出好的结果，单个用户能被路由离开该站点。

短下载被用于确定服务器的容量。名字服务器的分解延迟可通过这种测试，以及服务器开始发送数据所花的时间来确定。后一结果是强烈依赖于服务器负载，容量，性能。

长下载允许配置实用程序 34 确定包丢失，网络的拥挤，以及服务器应用如何影响文件的发送。确定前面的因子中那些基于单个用户的测试结果正使降低性能是不理想的。而且，这样引起的聚合，被存储在由 MSP32 维护的数据库中表示根段的情况。

应该指出，某些测试结果可同其它测试结果一起用。例如经容量

查询确定的传送站点上的负载能除以吞吐量测试的结果得到一平均期望下载时间，给出服务器的负载特性。

在所有指定的测试运行后，结果被收集及处理（步 48）。可能某些测试不能成功实现，在这种情况下，结果将指出一近似的最坏情况值（如零吞吐量或极高的延迟）

重要的是要指出，每次运行配置实用程序 34 时，并非所有可能测试都将被执行。当大量用户正使用系统时，通过单独的测试过程，将引起服务器和网络容量上的实质性消耗，增加了前面讨论的网络性能的螺旋式下降。

正如上面指出的，为了动态控制执行测试的用户数目，测试的频率数被存储在传送站点文件中。由配置实用程序 34 执行的测试被执行，以得到统计置信度，确定那一个传送站点最适合把数据传送给特定用户。统计置信度的获得或者是通过充分良好地测试小量用户样例，使用这些数据去影响传送站点选择，或者是通过大量用户，各自“轻微”测试几个可用站点整个使用那些数据。

因此，当系统开始使用时，相对小量的用户“登录”在系统上。由 MSP 32 维护的传送站点文件反映这些情况，并请求每个用户比较严格的测试网络（通过配置实用程序 34）。随着用户数的增加，传送站点文件被修改以减少每个用户执行的测试。这时，非常大量的用户正在使用该系统，配置实用程序 34 可能主要对可达到的传送站点测试（如上所述，通过“Ping”类型测试），以及首先依靠其它用户提供的并存储在由 MSP 32 维护的数据库中的测试数据。而且，即使当很多用户正在测试系统时，小量用户（如 1/5000）可被选择运行综合测试集。

更可取的是，测试应不提供大于近似总服务器负载的 5%。达到该目标的一方法是轻微测试大数量的服务器，产生一组具有足够性能的传送站点。这组传送站点则能轮流使用以检索数据。对组中每个传送站点的多媒体剪辑实际下载时间信息如下讨论的被累加，而传送站点性能的更多信息则能透明地提供给 MSP 32，不需进一步完全测试。

因此，根据这样收集的测试结果，及由 MSP 32 在传送站点文件

中提供的信息，配置实用程序 34 确定那一个传送站点或传送站点组对用户终端 12 是最佳的（步 12）。该判断能通过加权执行的不同测试和比较每个站点的得分而被数值化。

在当前的优选实施例中，为在相对小量传送站点和用户的在低通信量环境中使用，配置实用程序 34 对每个可用的传送站点首先进行 Ping(强制回应)和吞吐量测试。开始的强制回应测试被执行以确定是否传送站点可达到。在吞吐量测试中短的和长的下载被执行以确定来自传送站点的最大和最小吞吐量，并确定是否吞吐量变化小到足以适应视频数据传输。因此，这种测试是全部给以高权重。其它测试，如跟踪路由可被执行，并且报告给 MSP 32 的结果，不起选择传送站点的作用（这种其它的测试能给予零权重或接近零的权重）。随着系统尺寸的增大，及额外的传送站点和用户的加入，通过改变传送站点文件内容，站点选择的公式能被改变以反映网络条件改变。

在本发明的一优选实施例中，专用的图形界面被提供，以便用户的位置及每个测试站点的位置（几何的和电的两者）能被显示在连到用户终端 12 的监示器上，使站点之间的相对距离得以可视地指示。在该实施例中，显示器以“雷达屏”的形式显示，其上用户终端 12 及传送站点作为“雷达显示标志”叠加到有关几何区域的图上被显示。为鼓励用户使用应用及提供更多网络范围的数据，用户接口能允许用户进入附加性能测试的“ad-hoc”测试站点。这时，配置实用程序将测试缺省的 Web 页文件（如“index.html”）和用户要求的指定文件。来自用户选择的站点的分析结果被调整，以同其它站点来的结果形成合理的比较。

应该指出，本发明能维护传送站点的多个集，以适应几个内容提供商组。每个内容提供商只可能在某些站点上被镜象。因此，对具有唯一传送站点集的每个内容提供商，起初的传送站点由配置实用程序 34 选择。一旦前面的测试被运行，为达到这些目的，在该实施例中一数字的权重能被用于每个合适的传送站点集。大量的智能镜象站点被选择，为每个内容提供商选择一个。内容提供商组在传送站点文件中被

指定；每个可能的传送站点被标识属于一个或多个内容提供商组。当内容提供商组被使用时，能最少少到两个组；最大数实质上是不限的。

还应该指出，传送站点的区分优先的等级也能被产生和被维护。如果这被做了，为响应初始智能镜象站点的失败将允许系统退到下一个最高等级的智能镜象站点。

在智能镜象站点被选择后，某些数据经电子邮件或其它因特网电子协议将被发送给 MSP 32（步 52）。通过询问用户收到及选择的智能镜象站点的身份，以及所有的未加工的测试数据和结果，包括每个测试曾经运行的时间和日期的信息，被编译成文本文件（在该实施例中它被加密）。通过 MSP 32 接收，数据被存储在数据库中以便管理和分析系统使用。

最后，配置实用程序 34 为每个传送站点集或优先等级表，保存所选的智能镜象站点的身份到（加密的）配置文件（步 54）。配置实用程序也可能为每个测试过的传送站点保存有关性能的信息。客户机程序 36 使用加密的配置文件从适当的智能镜象站点下载数据文件（视频剪辑或其它内容）。

应该指出，在系统操作中，MSP 32 实现了某些功能，MSP 32 维护传送站点表，根据需要添加和删除站点。MSP 32 也维护网络性能的数据库，包括从运行配置实用程序 34 的用户处经电子邮件或其它手段接收的信息。当大量的数据从众多的用户处被接收时，数据库能提供因特网及其部分的性能和其它特性的有价值的信息。得到这些信息的各种数据处理技术是已知的。

本发明所用的传送站点的位置最终由许多因子确定，包括市场考虑及价格/利益分析。而且，存储在 MSP 的数据库中的数据能确认位于因特网或其它网上给定位置的传送站点的实用程序。在本实施例中，服务器被置于每个主骨干上（由单个公司维护的因特网部分）及为大量用户服务的其它因特线路上，如由 Regional Bell Operating Companies ("RBOCS") 操作的主线路。在为大量用户服务或具有繁重视频传送通信流量的某些网络上，服务器能放到网络的入网点

（“POPs”）确保每个用户能访问到快速服务器。

一旦配置实用程序 34 已经运行，用户能用该系统，以允许和方便数据文件的接收，尤其是视频剪辑,音频剪辑,软件程序及其它内容。

当时间过去和因特网的用户区的使用模式改变时，用户可变得不满意与用户终端 12 有关的智能镜象站点的性能。如果发生这种情况，用户是自由再运行配置实用程序 34。这时，附加的传送站点可能已进入服务，或者一预先已存在的站点可能比前面所赋予的执行得更好。而且,如果播放程序 36 判断所选的智能镜象站点没有充分地执行时(如十次的尝试中已失败三次)，播放程序 36 能提示用户再运行配置实用程序 34。在本发明的其它实施例中测试和镜象的赋予自动地被运行，按智能镜象服务上每次文件请求或某些间隙次数，如每隔一次，每第 3 次，每第 10 次或每第 100 次。

在本发明的实施例中，智能镜象系统用于定位从其下载参照 Web 页的视频或音频剪辑的传送站点。在本实施例中，客户机程序可指或认定为“播放程序”。播放程序除了执行客户机程序 36 的功能外，还提供视频数据的检索和回放。通常，浏览器程序 38 在用户终端 12 上运行以观察 Web 内容。典型使用的浏览器程序包括 NCSA Mosaic, Netscape Navigator, 及 Microsoft Internet Explorer。浏览器程序 38 允许用户在因特网上不同 Web 站点之间热链接。

EMBED 标志被用于 HTML 文档中以表示哪一个 Web 页包含系统管理的内容。当浏览器程序 38 接收包含 EMBED 标志的 Web 页时，开始以标志为依据下载文件，且分析文件的类型。如果这个文件是由播放程序 36 处理的类型，如 MPEG，浏览器程序 38 起动播放程序 36。标志的内容则由浏览器程序 38 传到播放程序 36。

播放程序 36（图 1）由 MSP 32 方便的提供智能镜象服务。播放程序 36 的工作详细示于图 3。

播放程序首先分析 EMBED 标志，判断是否存在“SM”（智能镜象）参数（步 60）；这种参数的存在表示为智能镜象允许嵌入剪辑。与“SM”参数有关的数据指示了所要求的剪辑源于那里的具体的内容提

供商，以及具体的内容提供商使用的镜象服务组。

如果播放程序 36 确定以 EMBED 标志为依据由系统处理的视频剪辑或其它的内容（步 62），从内容提供商 22 嵌入的剪辑的传送被停止。然后，如果存在，播放程序 36 从 EMBED 语句中抽取访问控制或等级信息（步 64）。此等级信息同存储在用户终端 12 的配置文件中的参考等级比较（步 66）。如该剪辑的等级信息不存在，配置文件被询问，以判断是否未定等级的剪辑如下面定义的可被播放（步 68）。根据前面的信息，播放程序 36 将批准或拒绝观看所要的剪辑。

如果回放被批准，播放程序 36 试图在属于用户终端 12 的本地计算机上找出引用的剪辑（步 70）。如果它存在，则不要重新下载，并且能直接在计算机上播放（从盘或从 RAM）（步 72）。然而，剪辑在本地计算机上生成的时间及日期首先与网络上可得到的剪辑的时间和日期核对，以确定是否存储的剪辑是最新的版本（步 74）。如果不是，存储的剪辑被遗弃（步 76）并如下进行下载处理。

如果在本地计算机上不存在剪辑，播放程序用下面形式创建一新的 URL（步 78）：“ http:// ”加上选择的存储在配置文件中的智能镜象站点的 IP 地址，加上到镜象文件的路径名（如“/pub/mirror/”），加上在EMBED语句中取自“SM”参数的内容提供商名，加上自 EMBED 语句得到的文件名。构成的 URL 用于从配置实用程序 34 选择的合适的智能镜象站点，检索选择的剪辑 (步 80)。如果对不同内容提供商存在超过一个传送站点集，“SM” 参数进一步被播放程序 36 使用，以确定配置文件中哪一个智能镜象站点被用于构成 URL（步 82）。在本发明的实施例中，站点选择至少部分通过重定向服务器实现。该实施例将参考图 4 详述如下。

如果对应于构成 URL 的剪辑在智能镜象站点没有被找到，或不能被访问，则从配置文件中下一个最高等级的智能镜象站点（步 84），或由重定向服务器选择的最高等级的传送站点进行下载处理（见图 4）。如所有传送站点失败，则由 EMBED 语句直接指定的原始内容提供商站点进行下载处理。

如回放不被允许，播放器阻止剪辑被传输（步 88）并显示一位图（步 90）忠告用户不允许下载。

如果播放程序 36 确定 EMBED 标志涉及的视频剪辑或其它内容未被系统操纵，播放者将检查是否在配置文件中的访问控制等级集允许用户播放这些被认为是“未定等级”的剪辑或者其它文件（步 92）。如果是，剪辑借助传统手段从其原始内容提供商 22 处被传输（步 94），并且播放程序 36 显示此下载文件（步 96）。如果不是，播放程序阻止剪辑被传输（步 98），并显示一消息（步 100）忠告用户不允许下载。

因下载，代表所要剪辑的数据文件被存储在本地计算机上指定的区域内，通常在属于用户终端 12 的硬盘上（步 102）。在本实施例中，该数据区基于最近最少使用(least-recently-used)由播放程序 36 管理。也就是说，如果在数据区没有空间保持新的剪辑，最近最少使用（或观看）的剪辑或一些剪辑能被废弃以得到空间（步 104）。

在本发明的实施例中，客户机程序 36 能向 MSP 32 发送消息（步 106）以反映是否下载成功。该消息包括用户终端 12 的因特网地址，所选择的服务器集的身份，完成下载所用站点的因特网地址，所有失败站点的因特网地址，下载文件名及下载文件的时间。这些信息也能被 MSP 32 使用，以跟踪文件下载实时确定是否智能镜象站点有任何问题。

另一方面，客户机程序 36 能维护一个文件传送性能的小型本地数据库。而每次的下载应被计时。特别是，收集服务器开始发送请求的文件所花费的时间，数据传输率稳定性，以及传送的错误率这类信息。在某些间隔上（如每周地或每 100 次下载）包含累加的文件传输性能信息的消息，以及上面讨论的用户和服务器信息将（自动地或根据请求）送到 MSP 32（步 106），以更新 MSP 的数据库。这附加信息增加了 MSP 的网络性能的“知识”而并不招致任何额外的测试开销。

该数据在确定传送站点性能，为评估从传送站点提供商处购买的服务质量，以及对内容提供商证明服务的质量，支持系统的价格方面特别有价值。而且，可以看出，很多同样的信息能够通过运行配置实

用程序 34 的新用户而获得。

在本发明的实施例中，来自 MSP 32 的数据库的信息能用于为给定用户预测改进的传送站点，而不需要实际上运行配置实用程序去测试用户和一系列可能的传送站点之间的网络。尤其是包含在 MSP 32 的数据库中收集的网络数据，按照给定的因特网 IP 地址和一些不同传送站点之间的性能差进行分析。根据该分析，引出用户的 IP 地址和提供较好的数据传送性能的传送站点之间的相互关系。该相关数据被用于产生一在传送站点选择过程中能被 MSP 32 使用的查找表。

实际上，重定向服务器被包含在 MSP 网络中。在本实施例中，重定向服务器的功能由 MSP 32 服务器在其中被执行，和实现。在另一实施例中，几个重定向服务器在整个网络的不同位置可被使用。

在操作中，当用户请求由 MSP 发送系统管理文件时，重定向服务器将取得用户的 IP 地址。如上讨论的，一请求通过 EMBED 语句以重定向服务构成。EMBED 语句能显式访问指定的服务器 IP 地址，识别单个重定向服务器（如 MSP 32），或能包含一脚本或可执行代码（如 JavaScript 或 Java 编程语言）指定很多重定向服务器，对其的访问可以顺序或随机的次序进行。而后，用户的 IP 地址由服务器通过传统方式确定，如典型地包含请求者地址信息的 HTTP（超文本传输协议）请求。接着，重定向服务器把用户的 IP 地址映射到位于查找表中的最佳传送站点，及把传送站点的地址转发给用户。用户的客户机程序 36 或重定向服务器则重定向文件请求，使文件从最佳传送站点传送。

设计查找表过程中的一个重要因子从因特网上大量的地址中得到。连到因特网上的每台计算机被赋给一个地址。在传统的因特网定址方案中，一个地址包含 4 个字节的值。通过转换，转换每个字节成一个十进制数（0-255）及用一个句点分隔各字节来表示。例如对 www.intervu.net服务器的地址是 192.215.147.185。用该地址方案，在因特网上有超过 4 百万个可能的地址。当前技术给定的包括 4 百万地址的查找表，在这里描述的实施例中使用是太大了，其方法是寻求压缩查找表。

已发现最佳传送站点的选择经常同用户 IP 地址的第一个字节有关。该关系是这样的，在运行了配置实用程序 34 以后，具有同样的第一字节地址的统计上有效数目的用户，将选择同样的传送站点，或出自一小传送站点组中的其它站点。显然，这些传送站点同无关连的传送站点比较，为具有相同的第一字节地址的大部分用户提供改善了的性能。在本发明的实施例中，这关系是足够明显的，形成的压缩查找表，包括数值为 0-255 的一系列第一字节 IP 地址，且对每个地址，一系列的传送站点为具有相应 IP 地址的用户提供改进的性能。这样，一个或多个传送站点被映射到地址表的每个入口。

应该指出，在某些情况下在查找表中的某些第一字节的 IP 地址可不映射到相应的服务器，例如当很少的用户具有对应于表中特定入口的 IP 地址，并且它们未能充分运行配置程序 34 以产生统计有效的或可靠的结果。在这种情况下，当一指定的服务器在数据表中没被标识时，一个或多个确省服务器可为用户在数据库中指定。

在一优选实施例中，查找表被存储在单个重定向服务器如 MSP 32 中。然而，查找表及相应的编程也能存储在 Web 页服务器，内容提供商 22 的服务器，或上面的组合中。

重定向服务器被访问，按下面和如图 4 所述，以选择传送站点（图 3 步 82）。用户经用户终端浏览器，通过双击具有那个文件的 EMBED 语句的链接请求一 Web 页引用的文件。如果文件请求是通过 MSP 的发送系统管理的文件，使用传统的 HTML 文件请求语句，将该文件请求转给重定向服务器（如 MSP 32）也就是服务器“GET”（步 120）。

重定向服务器检查输入的请求，并且使用由 Web 服务器提供的“REMOTE-HOST”变量确定用户的网络地址（IP 地址）（步 122）。应该指出，图 4 所说明的本发明的实施例中，无须用户终端 12 上安装的特定目的的客户机程序（如客户机程序 36）就能被完成。如果没有客户机程序 36 被安装（步 124），而且当用户被脚本或下载程序查询，不想安装客户机程序（步 126），则站点选择将完全通过重定向服务器实现。

然后，重定向服务器分析用户的 IP 地址，检查在查找表上一系列可能的传送站点，以确定哪个传送站点或哪些站点同用户的 IP 地址相应（步 128）。如果多于一个地址相应于用户查找表入口（步 130），重定向服务器则选择一传送站点地址。如果单个传送站点地址被找到，则 HTTP 重定向响应用于从所选的传送站点发送文件（步 132）。按照 HTTP 的规范，从某个服务器请求的文件通过此服务器能被重定向到其它位置，无需任何用户或客户机程序干涉。然后站点选择被完成（步 134）。为完成事务，客户机程序请求从所选的传送站点拷贝文件，及传送站点服务器检索文件并把它发送给用户。

如果用户想要安装客户机程序 36（步 126），则程序将通过传统手段下载和安装（步 136）。如果客户机程序 36 已被安装（步 124），或如果安装刚被完成，文件请求结果不同，并能既利用服务器侧又利用客户机侧以可能最有效的方法处理检索文件。

首先，重定向服务器分析用户的 IP 地址，并检查在查找表上可能的一系列传送站点，以确定哪个传送站点或哪些站点同用户的 IP 地址相应（步 138）。在某些情况下，存储查找表的重定向服务器可能将生成（步 140）和向用户终端 12 发送（步 142）包括传送站点子表的一小文件，并且依靠客户机程序 36 做出最终的传送站点选择（步 144）。这可以出现很多原因。

首先，如上所指出的，收集的网络性能数据的分析可能表明一些传送站点（与单个传送站点相反）对具有相同第一字节地址范围的某些用户提供改进的性能。这发生在例如，有很多用户，在运行配置实用程序后选择了不同的传送站点，全部具有相同的第一字节的地址或处在同一个第一字节地址内的地址范围。在这种情况下，客户机程序 36 能够取得通过重定向服务器下载的子表，并同镜象站点保存的表比较，该镜象站点是前面通过运行配置实用程序已选择的（见图 3，步 54）。如果能够匹配，客户机程序重定向文件请求到匹配的传送站点（步 146）。客户也应有选项以忽略由客户机程序做出的选择，并且从包含在查找表内的传送站点中做出最佳推测选择，因为一个或多个这样的

传送站点，可提供超过前面通过配置实用程序选择的任何传送站点改进的性能，尤其是如果配置实用程序最近没有被运行。

其次，情况也可能产生，那里由 MSP 管理的传送站点数及 IP 地址已增加以致查找表变得不切实际地太大，即重定向服务器在响应一新文件请求时有延迟，因为它正代表前面的文件请求忙于从头至尾检索查找表。如果在这种情况下，从头至尾检索查找表的任务在重定向服务器和客户机程序 36 之间被分开。因而用户终端 12 被要求作某些处理，并且通过作出最终传送站点的选择亦起路由器的作用。

为完成这些，重定向服务器根据给定的地址范围把查找表再分割成更小的子表。因此，当重定向服务器接收到文件请求时，服务器将把用户的 IP 地址（步 138）映射到相应地址范围的子表中，然后生成（步 140）并下载（步 142）给用户包含子表的小文件。而后客户机程序 36 起路由器的作用，从子查找表的传送站点中选择传送站点（步 144），或者把用户的 IP 地址映射到子表上相关的传送站点的合适地址，或者通过寻找子表上的传送站点和如上所述，在前面运行网络性能测试后，由配置实用程序 34 保存的优选站点表上的传送站点之间的匹配。一旦客户机程序已选择一传送站点，客户机程序重定向文件请求到所选择的传送站点（步 146）。通过重定向服务器和用户终端 12 之间的分开工作，文件请求能较快被处理，因此减少了用户请求文件的时间和当文件被用户终端 12 接收时的时间之间的延迟。

尽管 MSP 32 重定向查找表将根据网络性能的变化而频繁更新，被传送到用户的子表希望满足至少一些日子。因此，客户机程序 36 在获得新子表之前能重新使用该表一些日子。这些功能例如通过在 Web 页中嵌入脚本主管请求语句来实现。脚本能向客户机程序查询存储在用户终端 12 上编码于子表的失效日期。失效日期和子表的类型能被传回给重定向服务器，并且如果需要，服务器将传给一新的子表。

有可能特定的用户可请求由 MSP 发送系统管理的文件，而系统并不具备那个用户的知识（即，在用户的 IP 地址和查找表中的传送站点之间没找到匹配）。在这种情况下，重定向服务器为用户（步 130 和 144）

从重定向服务器上存储的缺省传送表中选择传送站点。当用户从该传送站点下载请求文件时，他可经嵌入的脚本或程序被提示，以改进内容传送为目的，从 MSP 32 获得配置实用程序文件，或者临时赋给一近似的服务器（如，类似但不相同的第一字节 IP 地址）以改进内容传送。如果用户获得和运行该配置实用程序，用户终端 12，如上所述，开始向 MSP 32 提供各种网络测试的结果，且这些信息将最后变为网络性能数据的 MSP 数据库的一部分。因此，对于新用户来的下一个文件请求，重定向服务器将能够根据额外的网络性能数据的分析，把该用户的 IP 地址映射到更合适的服务器。

通过修改查找表，重定向服务器也能就文件请求实现负载平衡和管理。如果 MSP 32（或单个控制的 MSP 32）预先确定部分的网络将停机一段时间，或者某些传送站点必须停机一段时间，查找表能被修改，以使重定向服务器将指点用户改换传送站点。因此，有可能两次紧接着地请求同一文件，一个用户可以被指向两个不同的传送站点。

应该理解，包含有查找表子表的文件能起传送站点文件的作用，由配置实用程序 34 使用以运行网络测试。子表表示了为改进性能已经过筛选的传送站点，该筛选是通过聚集从前面已实施过网络测试的一组其它用户处得到的网络性能数据进行的。传送站点的子表可通过运行配置实用程序 34 被进一步减肥或排定优先次序，以防该表及测试的网络次序是从前面的传送站点文件遗留的。一旦子表被排定优先次序，配置实用程序 34 如上所说的把表保存到（已加密的）配置文件中。

在下面所说的例子中，智能镜象系统利用查找表定位改进的传送站点，从其下载 Web 页上以广告标题为依据的视频剪辑。 当处理相对不受欢迎的内容如广告时，重要的是要尽可能快地向的用户传送广告内容，以便潜在的用户不失去兴趣。关于 Web 页上广告标题的内容，同样重要的是要尽可能快的向用户传送视频内容，不必事先下载大量实质性的软件。因此，作为另外一种利用本发明的优质服务，服务提供商通过因特网在 Web 页上能分发广告标题。每个标题将涉及一个视频剪辑，所有这些剪辑将存储在为此目的建立的内容提供商的服务器

上。

在该例中，客户机程序 36 被称为“播放程序”，并且它具有如前所说的附加的检索和回放视频数据的功能。浏览器 38（如 Netscape Navigator 或 Microsoft Internet Explorer）被用来观察用户终端 12 上的 Web 内容，并在用户终端 12 和因特网 10 上的其它计算机之间通信。

当客户机选择了在 Web 页上显示的广告标题时，编码在标题中的 EMBED 标志被浏览器检查。浏览器以 EMBED 标志为依据开始文件下载，及分析文件的类型。如果文件是播放程序处理的类型，如 MPEG，浏览器把标志传送给播放程序并起动播放程序。播放程序为视频剪辑的文件名检查 EMBED 标志，并试图在属于用户终端的本地计算机上查找此剪辑。 如果在那里它存在，则不重新下载，并且能直接在计算机上被播放（从盘或从 RAM）。如果视频剪辑不在本地计算机上，播放程序分析 EMBED 标记确定是否有“SM”(智能镜象) 参数；该参数的存在表示嵌入的剪辑对智能镜象是允许的。SM 参数进一步指出重定向服务器的地址，还有，如上所述，所要的剪辑源从特定内容提供商，以及特定内容提供商使用镜象服务器组。

如果播放程序识别出指出重定向服务器的参数，播放程序将调用浏览器，经传统的 HTML 文件请求技术，从重定向服务器请求的广告标题文件。重定向服务器检查进入的请求并确定客户机的 IP 地址。重定向服务器把客户机的 IP 地址映射到查找表及检索一传送站点地址或子表。传送站点地址（或传送站点子表以及 IP 地址）通过重定向服务器返回到播放程序。如果子表被返回，播放程序如前面所述，从所列的表中选择单个传送站点。一旦单个传送站点被确定，播放程序创建一新的 URL，其形式为：“http:// ”加上所选传送站点的 IP 地址，加上镜象文件路径名（如“/pub/mirror/”），加上根据 EMBED 语句中的 “SM” 参数取得的内容提供商名。加上从 EMBED 语句中取得的文件名。所构建的 URL 被用于从查找表选择的传送站点检索所选择的剪辑。

如果在 EMBED 标志中的 SM 参数没有指定重定向服务器，或如果重定向服务器不可得到并且不返回响应，播放程序将从存储在配置实用程序文件中的智能镜象站点的优先次序表中选择一 IP 地址，并替代 URL 中的该地址。如果播放程序没有检测出存在智能镜象站点优先次序表（例如，如果用户从未下载过配置实用程序 34），则播放程序使用初始内容提供商的服务器地址，就象直接被 EMBED 语句指定的，以请求视频。

在不是专门以广告内容传送为目标的应用中，提供下载信息给 MSP 便于使用本发明作为高级的基于预约的服务。随着成功的下载在数据库中被跟踪，每个用户能有一个相关的跟踪费用的 “帐号”。 能通过文件，兆字节，月，或其它已知手段，为智能镜象系统的使用向用户收费。在本实施例中，同文件相关的 EMBED 标志包括文件的付帐信息，或“价格”。如果发现下载是相当困难或缓慢，本发明的下载性能的跟踪能发布折扣和信贷。

为确保存储在智能镜象站点上的文件只给本发明授权的用户使用（例如那些已付帐的用户），存储在传送站点的文件能可任选加密的形式，并且上面所述的下载步骤能包括解密步骤。这种加密和解密能通过熟知的手段实现。

如上所说，由本发明管理的剪辑能具有与其相关的内容级别信息。这可从对应于剪辑的 EMBED 语句中提供的“PG”参数完成。在本实施例中由四个特性被评级：裸体，性，亵渎，暴力。因此，PG 参数能由四位参数指定。每个特性按一到三的数值被评定。1 对应于不滤波（即所有内容被允许），2 对应于某些滤波（如等于通常在广播电视中允许的级别），及 3 对应于最彻底的滤波（如对小孩）。包含在 EMBED 语句中文件的级别等级，同前面授权处理中存储在用户终端 12 的配置文件中包含的评定滤波等级比较，并且只有授权的文件被转送。

鉴于上述情况，可以理解本发明的实施例可用于许多不同应用中，以允许获取和分析在给定用户和内容提供商或传送站点之间网络的性能数据。因此，尽管所说的实施例描述了在因特网范围内操作的系统，

以及使用因特网类型地址方案，可以看出这种系统能用于其它网络环境，包括但不限于公司的“企业内部网”。

此外，尽管图示说明的实施例主要用在视频发送系统中，应该清楚，按照本发明的系统同样能用于分配各种其它类型的计算机数据（如应用程序，数据库文件，及其它商业信息，虚拟现实文件，多媒体如 Macromedia Shockware 文件，及如书本那样的大文本文件）。这些其它类型的数据如上面所详细讨论的按不同内容提供商组可由本发明管理；不同类型的程序（而不是播放程序 36）典型地应在用户终端 12 被调用去观察或使用其它类型的数据。

也应该指出，在用户终端 12 实现的所述的某些功能（特别是，由配置实用程序 34，或客户机/播放程序 36）能作为一独立程序被实现，如在浏览器程序内运行的“插件(plug-in)”或“helper application”，或从传送站点下载在浏览器环境中运行的 Java 小应用程序。为了使用户终端能运行 Microsoft Windows 操作系统，也使用称为 Microsoft ActiveX 的环境。

尽管某些示例的结构和操作已被说明，本发明并不受此限制，其应用范围根据下面陈述的权利要求书确定。

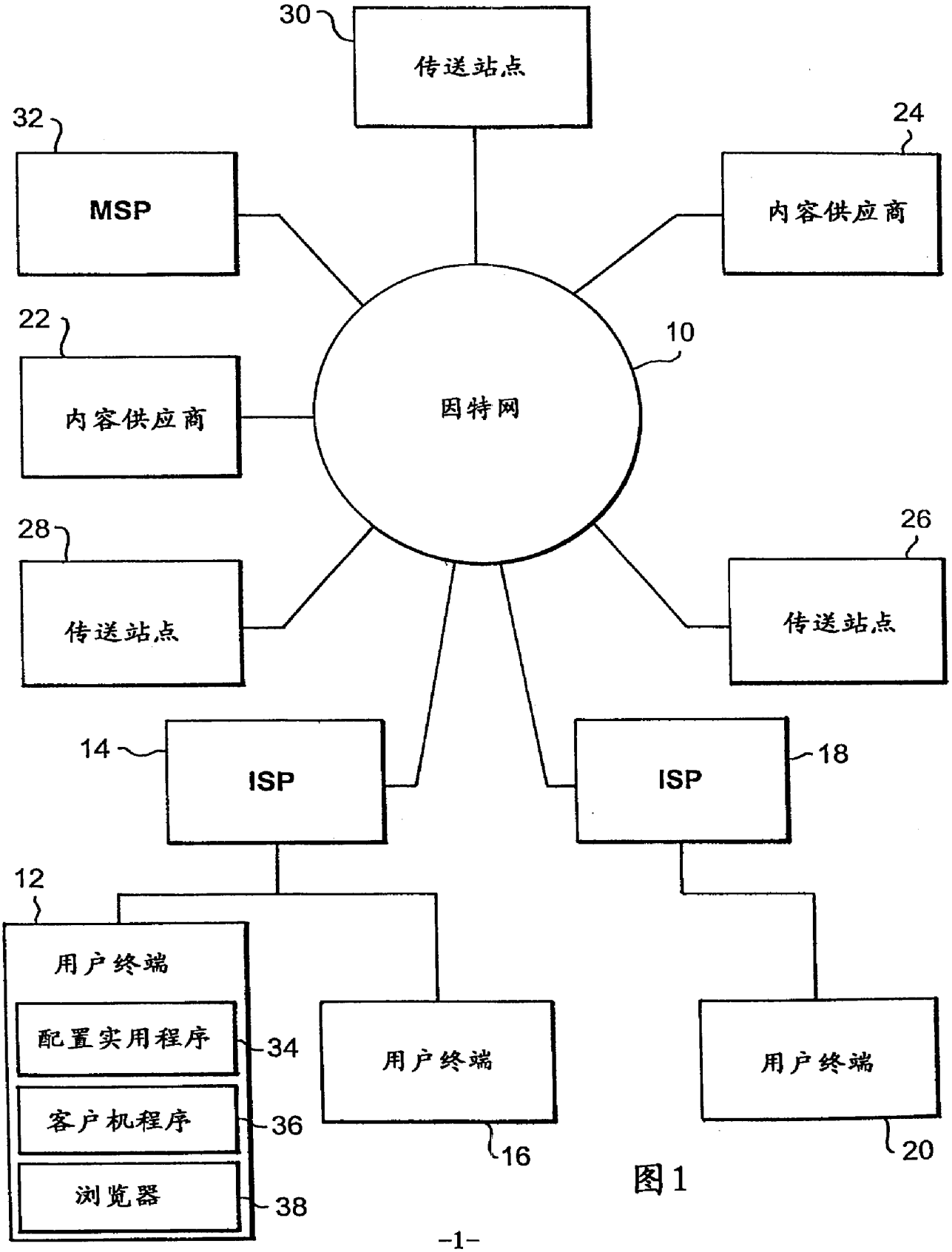

30
传送站点
32
MSP
24
内容供应商
22
内容供应商
10
因特网
28
传送站点
26
传送站点
14
ISP
18
ISP
12
用户终端
配置实用程序
34
客户机程序
36
浏览器
38
用户终端
16
用户终端
20

图1

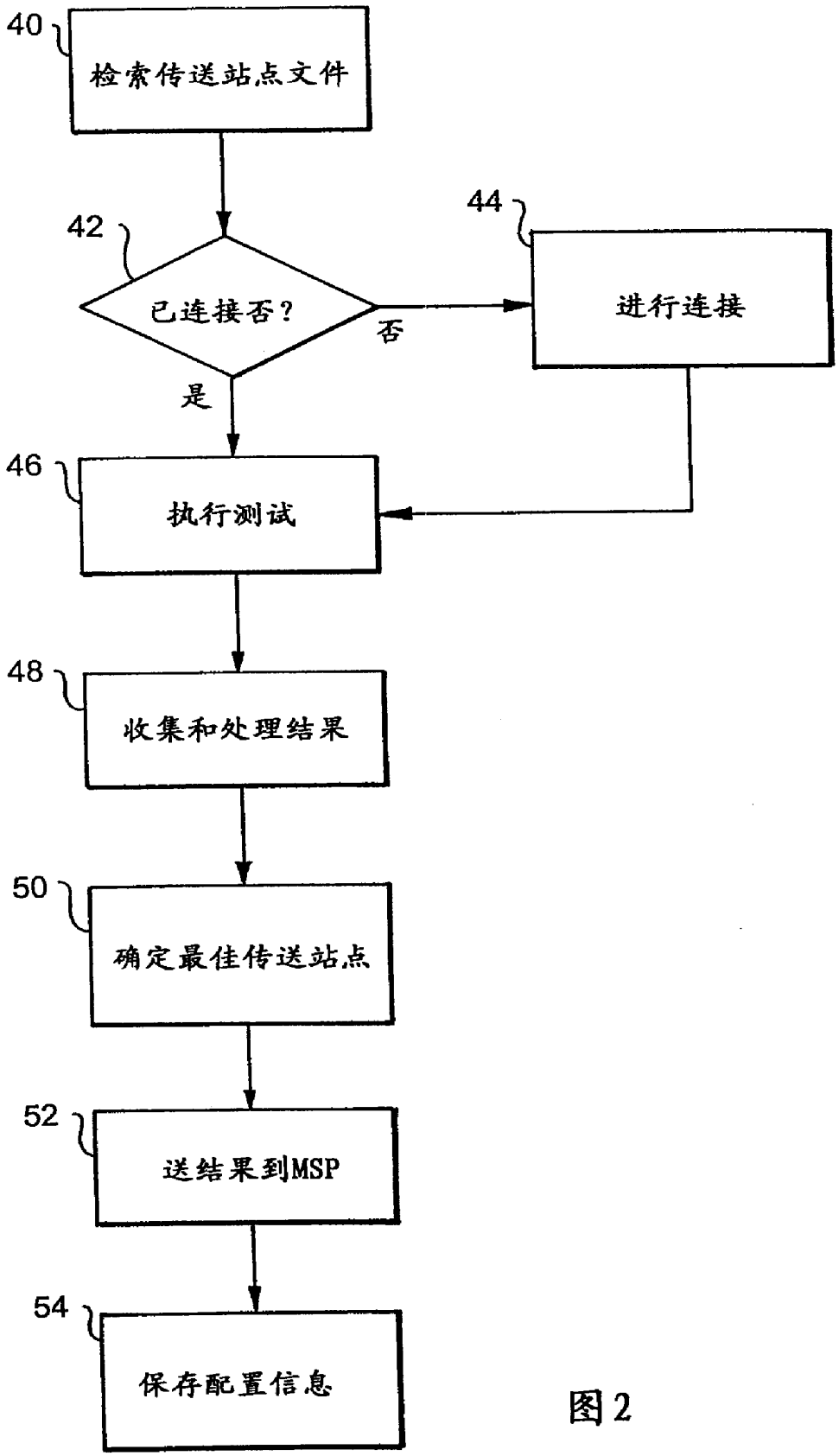

40
检索传送站点文件
42
已连接否？
否
是
44
进行连接
46
执行测试
48
收集和处理结果
50
确定最佳传送站点
52
送结果到MSP
54
保存配置信息

图 2

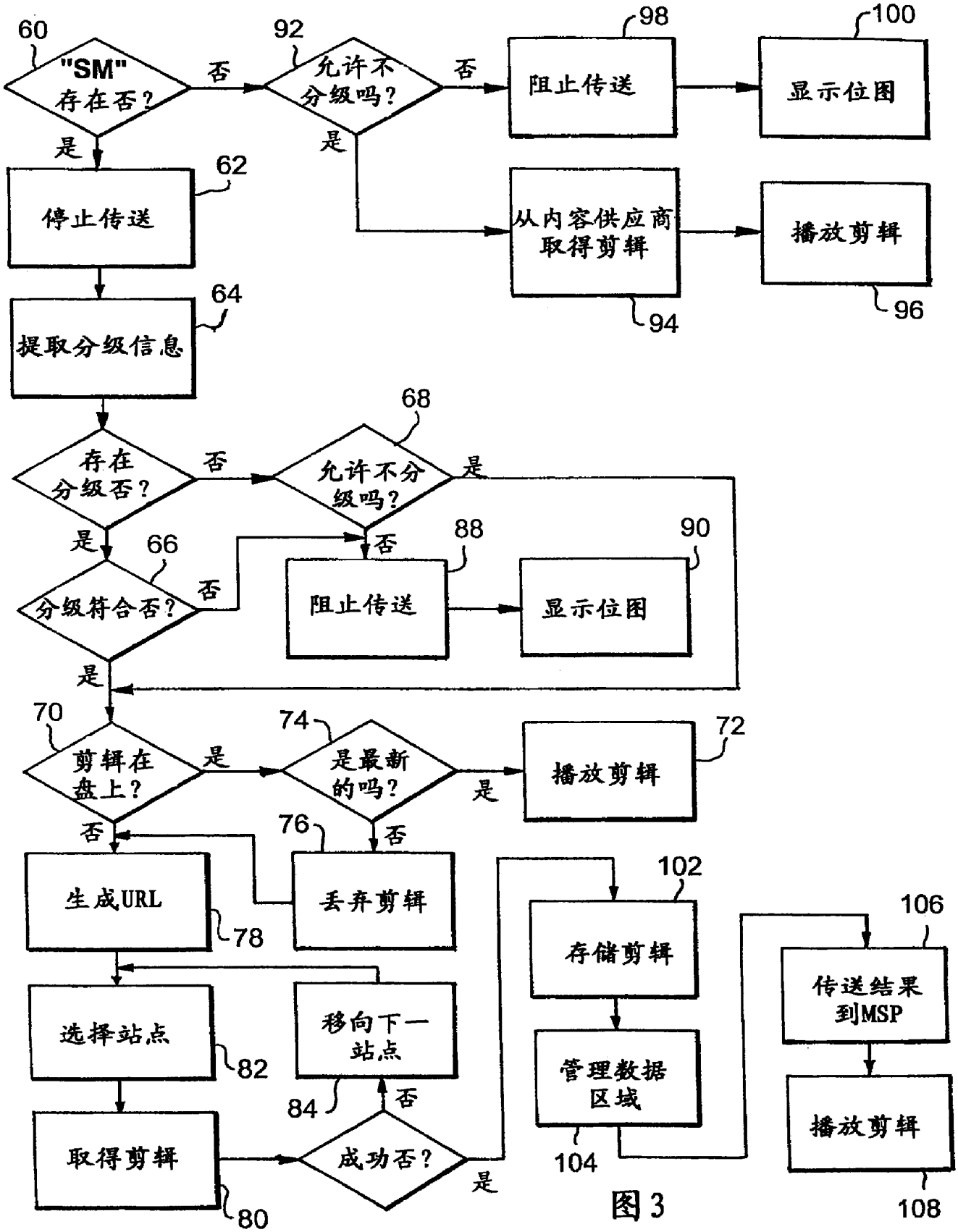

60
"SM"
存在否?
否
92
允许不
分级吗?
否
98
阻止传送
100
显示位图
是
是
62
停止传送
从内容供应商
取得剪辑
播放剪辑
94
96
64
提取分级信息
68
存在
分级否?
否
允许不分
级吗?
是
是
66
否
88
90
分级符合否?
否
阻止传送
显示位图
是
70
剪辑在
盘上?
是
74
是最新
的吗?
是
播放剪辑
72
否
76
否
生成URL
丢弃剪辑
78
102
存储剪辑
106
传送结果
到MSP
选择站点
移向下一
站点
82
84
否
管理数据
区域
播放剪辑
取得剪辑
成功否?
是
104
80
108

图 3

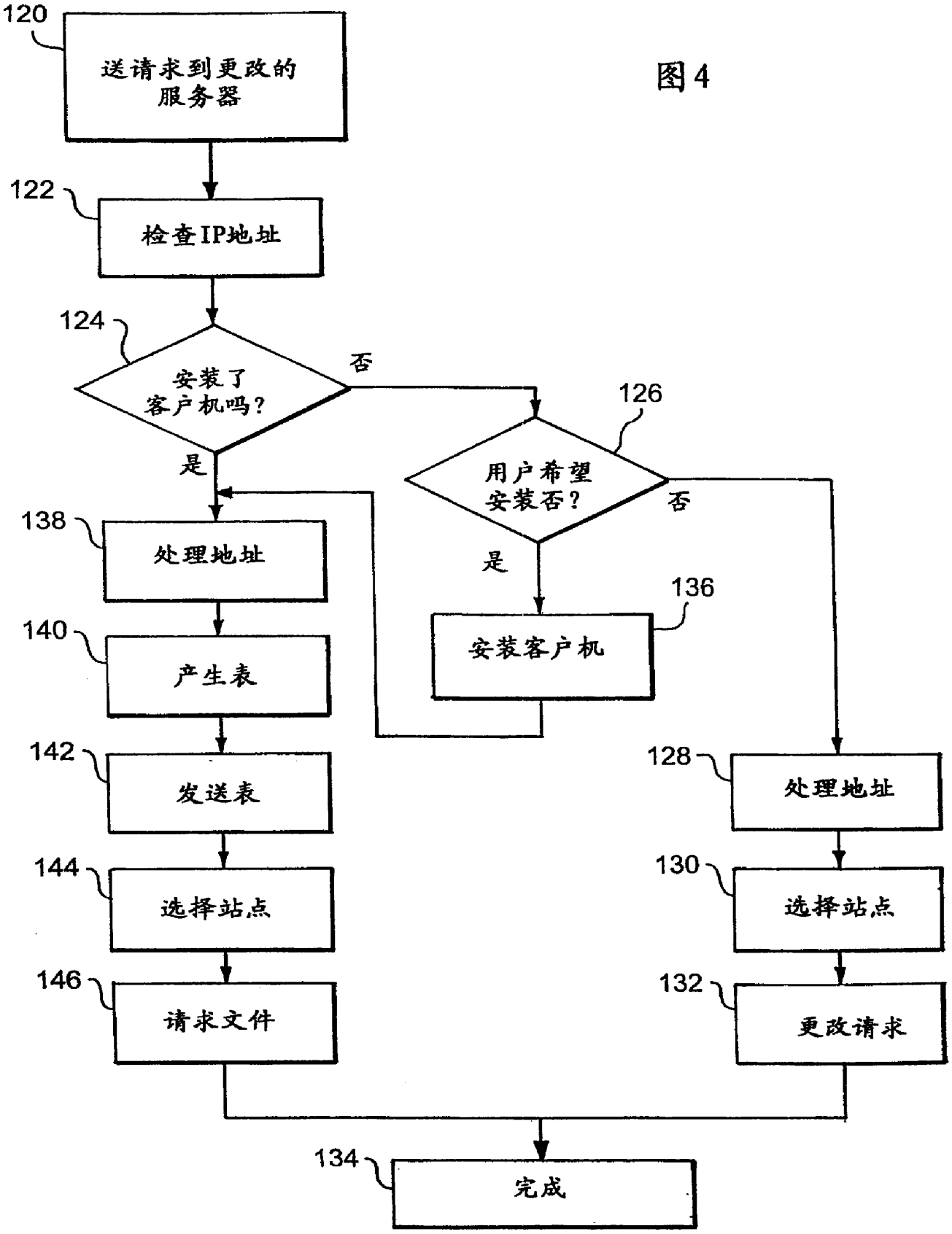

120
送请求到更改的服务器
122
检查IP地址
124
安装了客户机吗？
否
是
126
用户希望安装否？
否
是
138
处理地址
140
产生表
142
发送表
144
选择站点
146
请求文件
136
安装客户机
128
处理地址
130
选择站点
132
更改请求
134
完成

图4

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F — A2

(11) International Publication Number: **WO 96/42041**

(43) International Publication Date: 27 December 1996 (27.12.96)

(21) International Application Number: PCT/US96/07838

(22) International Filing Date: 3 June 1996 (03.06.96)

(30) Priority Data:

08/474,096	7 June 1995 (07.06.95)	US
08/486,797	7 June 1995 (07.06.95)	US

(81) Designated States: AU, CA, DE, GB, IL, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

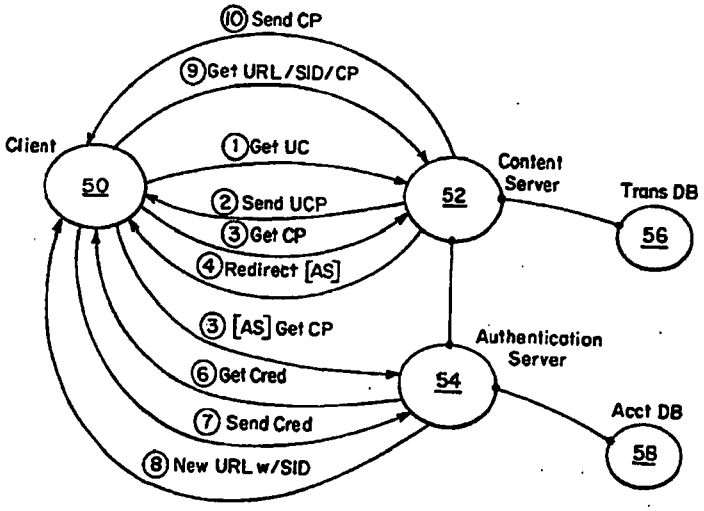
Published
Without international search report and to be republished upon receipt of that report.

(71) Applicant: OPEN MARKET, INC. [US/US]; 215 First Street, Cambridge, MA 02142 (US).

(72) Inventors: LEVERGOOD, Thomas, Mark; 9 North Street, Hopkinton, MA 01748 (US). STEWART, Lawrence, C.; 1 Arborwood Drive, Burlington, MA 01803 (US). MORRIS, Stephen, Jeffrey; 3 Kings Pine Road, Westford, MA 01886 (US). PAYNE, Andrew, C.; 5 Lewis Street, Lincoln, MA 01773 (US). TREESE, George, Winfield; 81 Saco Street, Newton, MA 02164 (US). GIFFORD, David, K.; 26 Pigeon Hill Road, Weston, MA 02193 (US).

(74) Agents: SMITH, James, M. et al.; Hamilton, Brook, Smith & Reynolds, Two Militia Drive, Lexington, MA 02173 (US).

(54) Title: INTERNET SERVER ACCESS CONTROL AND MONITORING SYSTEMS



(57) Abstract

This invention relates to methods for controlling and monitoring access to network servers. In particular, the process described in the invention includes client-server sessions over the Internet involving hypertext files. In the hypertext environment, a client views a document transmitted by a content server with a standard program known as the browser. Each hypertext document or page contains links to other hypertext pages which the user may select to traverse. When the user selects a link that is directed to an access-controlled file, the server subjects the request to a secondary server which determines whether the client has an authorization or valid account. Upon such verification, the user is provided with a session identification which allows the user to access to the requested file as well as any other files within the present protection domain.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	**GB**	United Kingdom	**MW**	Malawi
AT	Austria	**GE**	Georgia	**MX**	Mexico
AU	Australia	**GN**	Guinea	**NE**	Niger
BB	Barbados	**GR**	Greece	**NL**	Netherlands
BE	Belgium	**HU**	Hungary	**NO**	Norway
BF	Burkina Faso	**IE**	Ireland	**NZ**	New Zealand
BG	Bulgaria	**IT**	Italy	**PL**	Poland
BJ	Benin	**JP**	Japan	**PT**	Portugal
BR	Brazil	**KE**	Kenya	**RO**	Romania
BY	Belarus	**KG**	Kyrgystan	**RU**	Russian Federation
CA	Canada	**KP**	Democratic People's Republic of Korea	**SD**	Sudan
CF	Central African Republic			**SE**	Sweden
CG	Congo	**KR**	Republic of Korea	**SG**	Singapore
CH	Switzerland	**KZ**	Kazakhstan	**SI**	Slovenia
CI	Côte d'Ivoire	**LI**	Liechtenstein	**SK**	Slovakia
CM	Cameroon	**LK**	Sri Lanka	**SN**	Senegal
CN	China	**LR**	Liberia	**SZ**	Swaziland
CS	Czechoslovakia	**LT**	Lithuania	**TD**	Chad
CZ	Czech Republic	**LU**	Luxembourg	**TG**	Togo
DE	Germany	**LV**	Latvia	**TJ**	Tajikistan
DK	Denmark	**MC**	Monaco	**TT**	Trinidad and Tobago
EE	Estonia	**MD**	Republic of Moldova	**UA**	Ukraine
ES	Spain	**MG**	Madagascar	**UG**	Uganda
FI	Finland	**ML**	Mali	**US**	United States of America
FR	France	**MN**	Mongolia	**UZ**	Uzbekistan
GA	Gabon	**MR**	Mauritania	**VN**	Viet Nam

INTERNET SERVER ACCESS CONTROL AND MONITORING SYSTEMS

Reference to Appendix

A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by any one of the patent disclosure, as it appears in the Patent and Trademark Office patent files or records, but otherwise reserves all copyright rights whatsoever.

Background of the Invention

The Internet, which started in the late 1960s, is a vast computer network consisting of many smaller networks that span the entire globe. The Internet has grown exponentially, and millions of users ranging from individuals to corporations now use permanent and dial-up connections to use the Internet on a daily basis worldwide. The computers or networks of computers connected within the Internet, known as "hosts", allow public access to databases featuring information in nearly every field of expertise and are supported by entities ranging from universities and government to many commercial organizations.

The information on the Internet is made available to the public through "servers". A server is a system running on an Internet host for making available files or documents contained within that host. Such files are typically stored on magnetic storage devices, such as tape drives or fixed disks, local to the host. An Internet server may distribute information to any computer that requests the files on a host. The computer making such a request is known as the "client", which may be an Internet-connected

workstation, bulletin board system or home personal computer (PC).

TCP/IP (Transmission Control Protocol/Internet Protocol) is one networking protocol that permits full use of the Internet. All computers on a TCP/IP network need unique ID codes. Therefore, each computer or host on the Internet is identified by a unique number code, known as the IP (Internet Protocol) number or address, and corresponding network and computer names. In the past, an Internet user gained access to its resources only by identifying the host computer and a path through directories within the host's storage to locate a requested file. Although various navigating tools have helped users to search resources on the Internet without knowing specific host addresses, these tools still require a substantial technical knowledge of the Internet.

The World-Wide Web (Web) is a method of accessing information on the Internet which allows a user to navigate the Internet resources intuitively, without IP addresses or other technical knowledge. The Web dispenses with command-line utilities which typically require a user to transmit sets of commands to communicate with an Internet server. Instead, the Web is made up of hundreds of thousands of interconnected "pages", or documents, which can be displayed on a computer monitor. The Web pages are provided by hosts running special servers. Software which runs these Web servers is relatively simple and is available on a wide range of computer platforms including PC's. Equally available is a form of client software, known as a Web "browser", which is used to display Web pages as well as traditional non-Web files on the client system. Today, the Internet hosts which provide Web servers are increasing at a rate of more than 300 per month, en route to becoming the preferred method of Internet communication.

Created in 1991, the Web is based on the concept of "hypertext" and a transfer method known as "HTTP" (Hypertext Transfer Protocol). HTTP is designed to run primarily over TCP/IP and uses the standard Internet setup, where a server issues the data and a client displays or processes it. One format for information transfer is to create documents using Hypertext Markup Language (HTML). HTML pages are made up of standard text as well as formatting codes which indicate how the page should be displayed. The Web client, a browser, reads these codes in order to display the page. The hypertext conventions and related functions of the world wide web are described in the appendices of U.S. Patent Application Serial No. 08/328,133, filed on October 24, 1994, by Payne et al. which is incorporated herein by reference.

Each Web page may contain pictures and sounds in addition to text. Hidden behind certain text, pictures or sounds are connections, known as "hypertext links" ("links"), to other pages within the same server or even on other computers within the Internet. For example, links may be visually displayed as words or phrases that may be underlined or displayed in a second color. Each link is directed to a web page by using a special name called a URL (Uniform Resource Locator). URLs enable a Web browser to go directly to any file held on any Web server. A user may also specify a known URL by writing it directly into the command line on a Web page to jump to another Web page.

The URL naming system consists of three parts: the transfer format, the host name of the machine that holds the file, and the path to the file. An example of a URL may be:

http://www.college.univ.edu/Adir/Bdir/Cdir/page.html,

where "http" represents the transfer protocol; a colon and two forward slashes (://) are used to separate the transfer format from the host name; "www.college.univ.edu" is the host name in which "www" denotes that the file being requested is a Web page; "/Adir/Bdir/Cdir" is a set of directory names in a tree structure, or a path, on the host machine; and "page.html" is the file name with an indication that the file is written in HTML.

The Internet maintains an open structure in which exchanges of information are made cost-free without restriction. The free access format inherent to the Internet, however, presents difficulties for those information providers requiring control over their Internet servers. Consider for example, a research organization that may want to make certain technical information available on its Internet server to a large group of colleagues around the globe, but the information must be kept confidential. Without means for identifying each client, the organization would not be able to provide information on the network on a confidential or preferential basis. In another situation, a company may want to provide highly specific service tips over its Internet server only to customers having service contracts or accounts.

Access control by an Internet server is difficult for at least two reasons. First, when a client sends a request for a file on a remote Internet server, that message is routed or relayed by a web of computers connected through the Internet until it reaches its destination host. The client does not necessarily know how its message reaches the server. At the same time, the server makes responses without ever knowing exactly who the client is or what its IP address is. While the server may be programmed to trace its clients, the task of tracing is often difficult, if not impossible. Secondly, to prevent unwanted intrusion into

private local area networks (LAN), system administrators implement various data-flow control mechanisms, such as the Internet "firewalls", within their networks. An Internet firewall allows a user to reach the Internet anonymously while preventing intruders of the outside world from accessing the user's LAN.

Summary of the Invention

The present invention relates to methods of processing service requests from a client to a server through a network. In particular the present invention is applicable to processing client requests in an HTTP (Hypertext Transfer Protocol) environment, such as the World-Wide Web (Web). One aspect of the invention involves forwarding a service request from the client to the server and appending a session identification (SID) to the request and to subsequent service requests from the client to the server within a session of requests. In a preferred embodiment, the present method involves returning the SID from the server to the client upon an initial service request made by the client. A valid SID may include an authorization identifier to allow a user to access controlled files.

In a preferred embodiment, a client request is made with a Uniform Resource Locator (URL) from a Web browser. Where a client request is directed to a controlled file without an SID, the Internet server subjects the client to an authorization routine prior to issuing the SID, the SID being protected from forgery. A content server initiates the authorization routine by redirecting the client's request to an authentication server which may be at a different host. Upon receiving a redirected request, the authentication server returns a response to interrogate the client and then issues an SID to a qualified client. For a new client, the authentication server may open a new account and issue an SID thereafter. A valid SID typically

comprises a user identifier, an accessible domain, a key identifier, an expiration time such as date, the IP address of the user computer, and an unforgettable digital signature such as a cryptographic hash of all of the other items in the SID encrypted with a secret key. The authentication server then forwards a new request consisting of the original URL appended by the SID to the client in a REDIRECT. The modified request formed by a new URL is automatically forwarded by the client browser to the content server.

When the content server receives a URL request accompanied by an SID, it logs the URL with the SID and the user IP address in a transaction log and proceeds to validate the SID. When the SID is so validated, the content server sends the requested document for display by the client's Web browser.

In the preferred embodiment, a valid SID allows the client to access all controlled files within a protection domain without requiring further authorization. A protection domain is defined by the service provider and is a collection of controlled files of common protection within one or more servers.

When a client accesses a controlled Web page with a valid SID, the user viewing the page may want to traverse a link to view another Web page. There are several possibilities. The user may traverse a link to another page in the same path. This is called a "relative link". A relative link may be made either within the same domain or to a different domain. The browser on the client computer executes a relative link by rewriting the current URL to replace the old controlled page name with a new one. The new URL retains all portions of the old, including the SID, except for the new page name. If the relative link points to a page in the same protection domain, the SID remains valid, and the request is honored. However, if the

relative link points to a controlled page in a different protection domain, the SID is no longer valid, and the client is automatically redirected to forward the rewritten URL to the authentication server to update the SID. The updated or new SID provides access to the new domain if the user is qualified.

The user may also elect to traverse a link to a document in a different path. This is called an "absolute link". In generating a new absolute link, the SID is overwritten by the browser. In the preferred embodiment, the content server, in each serving of a controlled Web page within the domain, filters the page to include the current SID in each absolute URL on the page. Hence, when the user elects to traverse an absolute link, the browser is facilitated with an authenticated URL which is directed with its SID to a page in a different path. In another embodiment, the content server may forego the filtering procedure as above-described and redirect an absolute URL to the authentication server for an update.

An absolute link may also be directed to a controlled file in a different domain. Again, such a request is redirected to the authentication server for processing of a new SID. An absolute link directed to an uncontrolled file is accorded an immediate access.

In another embodiment, a server access control may be maintained by programming the client browser to store an SID or a similar tag for use in each URL call to that particular server. This embodiment, however, requires a special browser which can handle such communications and is generally not suitable for the standard browser format common to the Web.

Another aspect of the invention is to monitor the frequency and duration of access to various pages both controlled and uncontrolled. A transaction log within a content server keeps a history of each client access to a

page including the link sequence through which the page was accessed. Additionally, the content server may count the client requests exclusive of repeated requests from a common client. Such records provide important marketing feedback including user demand, access pattern, and relationships between customer demographics and accessed pages and access patterns.

The above and other features of the invention including various novel details of construction and combinations of parts will now be more particularly described with reference to the accompanying drawings and pointed out in the claims. It will be understood that the particular devices and methods embodying the invention are shown by way of illustration only and not as limitations of the invention. The principles and features of this invention may be employed in varied and numerous embodiments without departing from the scope of the invention.

Brief Description of the Drawings:

Figure 1 is a diagram illustrating the Internet operation.

Figure 2A is a flowchart describing the preferred method of Internet server access control and monitoring.

Figure 2B is a related flowchart describing the details of the authentication process.

Figure 3 illustrates an example of a client-server exchange session involving the access control and monitoring method of the present invention.

Figure 4 is an example of a World Wide Web page.

Figure 5 is an example of an authorization form page.

Figure 6 is a diagram describing the details of the translation of telephone numbers to URLs.

Detailed Description of the Invention:

Referring now to the drawings, Figure 1 is a graphical illustration of the Internet. The Internet 10 is a network of millions of interconnected computers 12 including systems owned by Internet providers 16 and information systems (BBS) 20 such as Compuserve or America Online. Individual or corporate users may establish connections to the Internet in several ways. A user on a home PC 14 may purchase an account through the Internet provider 16. Using a modem 22, the PC user can dial up the Internet provider to connect to a high speed modem 24 which, in turn, provides a full service connection to the Internet. A user 18 may also make a somewhat limited connection to the Internet through a BBS 20 that provides an Internet gateway connection to its customers.

Figure 2A is a flowchart detailing the preferred process of the present invention and Figure 4 illustrates a sample Web page displayed at a client by a browser. The page includes text 404 which includes underlined link text 412. The title bar 408 and URL bar 402 display the title and URL of the current web page, respectively. As shown in Figure 4, the title of the page is "Content Home Page" and the corresponding URL is "http://content.com/homepage". When a cursor 414 is positioned over link text 412b, the page which would be retrieved by clicking a mouse is typically identified in a status bar 406 which shows the URL for that link. In this example the status bar 406 shows that the URL for the pointed link 412b is directed to a page called "advertisement" in a commercial content server called "content". By clicking on the link text, the user causes the browser to generate a URL GET request at 100 in Figure 2A. The browser forwards the request to a content server 120, which processes the request by first determining whether the requested page is a controlled document 102. If the request is directed to an

uncontrolled page, as in "advertisement" page in this example, the content server records the URL and the IP address, to the extent it is available, in the transaction log 114. The content server then sends the requested page to the browser 116 for display on the user computer 117.

If the request is directed to a controlled page, the content server determines whether the URL contains an SID 102. For example, a URL may be directed to a controlled page name "report", such as "http://content.com/report", that requires an SID. If no SID is present, as in this example, the content server sends a "REDIRECT" response 122 to the browser 100 to redirect the user's initial request to an authentication server 200 to obtain a valid SID. The details of the authentication process are described in Figure 2B and will be discussed later, but the result of the process is an SID provided from the authentication server to the client. In the above example, a modified URL appended with an SID may be: "http://content.com/[SID]/report". The preferred SID is a sixteen character ASCII string that encodes 96 bits of SID data, 6 bits per character. It contains a 32-bit digital signature, a 16-bit expiration date with a granularity of one hour, a 2-bit key identifier used for key management, an 8-bit domain comprising a set of information files to which the current SID authorizes access, and a 22-bit user identifier. The remaining bits are reserved for expansion. The digital signature is a cryptographic hash of the remaining items in the SID and the authorized IP address which are encrypted with a secret key which is shared by the authentication and content servers.

If the initial GET URL contains a SID, the content server determines whether the request is directed to a page within the current domain 106. If the request having a SID is directed to a controlled page of a different domain, the

SID is no longer valid and, again, the user is redirected to the authentication server 122.

If the request is for a controlled page within the current domain, the content server proceeds to log the request URL, tagged with SID, and the user IP address in the transaction log 108. The content server then validates the SID 110. Such validation includes the following list of checks: (1) the SID's digital signature is compared against the digital signature computed from the remaining items in the SID and the user IP address using the secret key shared by the authentication and content servers; (2) the domain field of the SID is checked to verify that it is within the domain authorized; and (3) the EXP field of the SID is checked to verify that it is later than the current time.

If the validation passes, the content server searches the page to be forwarded for any absolute URL links contained therein 112, that is, any links directed to controlled documents in different content servers. The content server augments each absolute URL with the current SID to facilitate authenticated accesses across multiple content servers. The requested page as processed is then transmitted to the client browser for display 117. The user viewing the requested Web page may elect to traverse any link on that page to trigger the entire sequence again 100.

Figure 2B describes the details of the authentication process. The content server may redirect the client to an authentication server. The REDIRECT URL might be: "http://auth.com/authenticate?domain=[domain]&URL=http://content.com/report". That URL requests authentication and specifies the domain and the initial URL. In response to the REDIRECT, the client browser automatically sends a GET request with the provided URL.

Whenever the content server redirects the client to the authentication server 200, the authentication server initiates the authorization process by validating that it is for an approved content server and determining the level of authentication required for the access requested 210. Depending on this level, the server may challenge the user 212 for credentials. If the request is for a low level document, the authentication may issue an appropriate SID immediately 228 and forego the credential check procedures. If the document requires credentials, the authentication server sends a "CHALLENGE" response which causes the client browser to prompt the user for credentials 214. A preferred credential query typically consists of a request for user name and password. If the user is unable to provide a password, the access is denied. The browser forms an authorization header 300 from the information provided, and resends a GET request to the authentication server using the last URL along with an authorization header. For example, a URL of such a GET request may be: "http://auth.com/authenticate?domain=[domain]&URL=http://content.com/report and the authorization header may be:"AUTHORIZE:[authorization]".

Upon receiving the GET request, the authentication server queries an account database 216 to determine whether the user is authorized 218 to access the requested document. A preferred account database may contain a user profile which includes information for identifying purposes, such as client IP address and password, as well as user demographic information, such as user age, home address, hobby, or occupation, for later use by the content server. If the user is authorized, an SID is generated 228 as previously described. If the user is not cleared for authorization, the authentication server checks to see if the user qualifies for a new account 220. If the user is not qualified to open a new account, a page denying access

222 is transmitted to the client browser 100. If the user is qualified, the new user is sent a form page such as illustrated in Figure 5 to initiate a real-time on-line registration 224. The form may, for example, require personal information and credit references from the user. The browser is able to transmit the data entered by the user in the blanks 502 as a "POST" message to the authentication server. A POST message causes form contents to be sent to the server in a data body other than as part of the URL. If the registration form filled out by the new user is valid 226, an appropriate SID is generated 228. If the registration is not valid, access is again denied 222.

An SID for an authorized user is appended ("tagged") 230 to the original URL directed to a controlled page on the content server. The authentication server then transmits a REDIRECT response 232 based on the tagged URL to the client browser 100. The modified URL, such as "http://content.com/[SID]/report" is automatically forwarded to the content server 120.

Figure 3, illustrates a typical client-server exchange involving the access control and monitoring method of the present invention. In Step 1, the client 50 running a browser transmits a GET request through a network for an uncontrolled page (UCP). For example, the user may request an advertisement page by transmitting a URL "http://content.com/advertisement", where "content.com" is the server name and "advertisement" is the uncontrolled page name. In Step 2, the content server 52 processes the GET request and transmits the requested page, "advertisement". The content server also logs the GET request in the transaction database 56 by recording the URL, the client IP address, and the current time.

In Step 3, the user on the client machine may elect to traverse a link in the advertisement page directed to a controlled page (CP). For example, the advertisement page

may contain a link to a controlled page called "report". Selecting this link causes the client browser 50 to forward a GET request through a URL which is associated with the report file "http://content.com/report". The content server 52 determines that the request is to a controlled page and that the URL does not contain an SID. In Step 4, the content server transmits a REDIRECT response to the client, and, in Step 5, the browser automatically sends the REDIRECT URL to the authentication server 54. The REDIRECT URL sent to the authentication server may contain the following string:
"http://auth.com/authenticate?domain=[domain]&URL=http://content.com/report"

The authentication server processes the REDIRECT and determines whether user credentials (CRED) are needed for authorization. In Step 6, the authentication server transmits a "CHALLENGE" response to the client. As previously described, typical credentials consist of user name and password. An authorization header based on the credential information is then forwarded by the client browser to the authentication server. For example, a GET URL having such an authorization header is:
"http://autho.com/authenticate?domain=[domain]&URL=http://content.com/report and the authorization header may be:"AUTHORIZE:[authorization]". The authentication server processes the GET request by checking the Account Database 58. If a valid account exists for the user, an SID is issued which authorizes access to the controlled page "report" and all the other pages within the domain.

As previously described, the preferred SID comprises a compact ASCII string that encodes a user identifier, the current domain, a key identifier, an expiration time, the client IP address, and an unforgeable digital signature. In Step 8, the authentication server redirects the client to the tagged URL, "http://content.com/[SID]/report", to

the client. In Step 9, the tagged URL is automatically forwarded by the browser as a GET request to the content server. The content server logs the GET request in the Transaction database 56 by recording the tagged URL, the client IP address, and the current time. In Step 10, the content server, upon validating the SID, transmits the requested controlled page "report" for display on the client browser.

According to one aspect of the present invention, the content server periodically evaluates the record contained in the transaction log 56 to determine the frequency and duration of accesses to the associated content server. The server counts requests to particular pages exclusive of repeated requests from a common client in order to determine the merits of the information on different pages for ratings purposes. By excluding repeated calls, the system avoids distortions by users attempting to "stuff the ballot box." In one embodiment, the time intervals between repeated requests by a common client are measured to exclude those requests falling within a defined period of time.

Additionally, the server may, at any given time, track access history within a client-server session. Such a history profile informs the service provider about link transversal frequencies and link paths followed by users. This profile is produced by filtering transaction logs from one or more servers to select only transactions involving a particular user ID (UID). Two subsequent entries, A and B, corresponding to requests from a given user in these logs represent a link traversal from document A to document B made by the user in question. This information may be used to identify the most popular links to a specific page and to suggest where to insert new links to provide more direct access. In another embodiment, the access history is evaluated to determine traversed links leading to a

purchase of a product made within commercial pages. This information may be used, for example, to charge for advertising based on the number of link traversals from an advertising page to a product page or based on the count of purchases resulting from a path including the advertisement. In this embodiment, the server can gauge the effectiveness of advertising by measuring the number of sales that resulted from a particular page, link, or path of links. The system can be configured to charge the merchant for an advertising page based on the number of sales that resulted from that page.

According to another aspect of the present invention, a secondary server, such as the authentication server 200 in Figure 2B, may access a prearranged user profile from the account database 216 and include information based on such a profile in the user identifier field of the SID. In a preferred embodiment, the content server may use such an SID to customize user requested pages to include personalized content based on the user identifier field of the SID.

In another aspect of the invention, the user may gain access to domain of servers containing journals or publications through a subscription. In such a situation, the user may purchase the subscription in advance to gain access to on-line documents through the Internet. The user gains access to a subscribed document over the Internet through the authorization procedure as described above where an authorization indicator is preferably embedded in a session identifier. In another embodiment, rather than relying on a prepaid subscription, a user may be charged and billed each time he or she accesses a particular document through the Internet. In that case, authorization may not be required so long as the user is fully identified in order to be charged for the service. The user

identification is most appropriately embedded in the session identifier described above.

In another aspect of the invention, facilities are provided to allow users to utilize conventional telephone numbers or other identifiers to access merchant services. These merchant services can optionally be protected using SIDs. In a preferred embodiment, as shown in Figure 6, a Web browser client 601 provides a "dial" command to accept a telephone number from a user, as by clicking on a "dial" icon and inputting the telephone number through the keyboard. The browser then constructs a URL of the form "http://directory.net/NUMBER", where NUMBER is the telephone number or other identifier specified by the user. The browser then performs a GET of the document specified by this URL, and contacts directory server 602, sending the NUMBER requested in Message 1.

In another embodiment, implemented with a conventional browser, client 601 uses a form page provided by directory server 601 that prompts for a telephone number or other identifier in place of a "dial" command, and Message 1 is a POST message to a URL specified by this form page.

Once NUMBER is received by directory server 601, the directory server uses database 604 to translate the NUMBER to a target URL that describes the merchant server and document that implements the service corresponding to NUMBER. This translation can ignore the punctuation of the number, therefore embedded parenthesis or dashes are not significant.

In another embodiment an identifier other than a number may be provided. For example, a user may enter a company name or product name without exact spelling. In such a case a "soundex" or other phonetic mapping can be used to permit words that sound alike to map to the same target URL. Multiple identifiers can also be used, such as

a telephone number in conjunction with a product name or extension.

In Message 2, Directory server 602 sends a REDIRECT to client 601, specifying the target URL for NUMBER as computed from database 604. The client browser 601 then automatically sends Message 3 to GET the contents of this URL. Merchant server 603 returns this information in Message 4. The server 602 might have returned a Web page to the client to provide an appropriate link to the required document. However, because server 602 makes a translation to a final URL and sends a REDIRECT rather than a page to client 601, the document of message 4 is obtained without any user action beyond the initial dial input.

The Target URL contained in Message 3 can be an ordinary URL to an uncontrolled page, or it can be a URL that describes a controlled page. If the Target URL describes a controlled page then authentication is performed as previously described. The Target URL can also describe a URL that includes an SID that provides a preauthorized means of accessing a controlled page.

Among benefits of the "dial" command and its implementation is an improved way of accessing the Internet that is compatible with conventional telephone numbers and other identifiers. Merchants do not need to alter their print or television advertising to provide an Internet specific form of contact information, and users do not need to learn about URLs.

In the approach a single merchant server can provide multiple services that correspond to different external "telephone numbers" or other identifiers. For example, if users dial the "flight arrival" number they could be directed to the URL for the arrival page, while, if they dial the "reservations" number, they would be directed to the URL for the reservations page. A "priority gold" number could be directed to a controlled page URL that

would first authenticate the user as belonging to the gold users group, and then would provide access to the "priority gold" page. An unpublished "ambassador" number could be directed to a tagged URL that permits access to the "priority gold" page without user authentication.

This invention has particular application to network sales systems such as presented in U.S. Patent Application Serial No. 08/328,133, filed October 24, 1994, by Payne et al. which is incorporated herein by reference.

Equivalents:

Those skilled in the art will know, or be able to ascertain using no more than routine experimentation, many equivalents to the specific embodiments or the invention described herein. These and all other equivalents are intended to be encompassed by the following claims.

Appendix

```
/* TclIdSid
 *    Scans an ascii line and finds an ascii SID. (no validation though)
 * Inputs:
 *    lineoftext
 * Returns:
 *    ascii bin_sid, if a sid is found it is returned.
 *
 */

int TclIdSid(ClientData dummy, Tcl_Interp *interp,
                          int argc, char **argv)
   {
   char *sidp, *cp;

   interp->result[0] = 0;

   if (argc != 2)
     {
     interp->result = "wrong # args";
     return TCL_ERROR;
     }
   sidp = (char *) strstr(argv[1], "/@@");
   if (sidp == NULL) return TCL_OK;
   cp = (char *) strstr(sidp+1,"/");
   if ((cp == NULL) && (strlen(sidp) != 19)) return TCL_OK;
   if ((cp - sidp) != 19) return TCL_OK;
   strncpy(interp->result, sidp,19);
   interp->result[19] = 0;
   return TCL_OK;
   }


/*
 * Register commands with interpreter.
 */
int SidSupInit(Tcl_Interp *interp)
  {
  Tcl_CreateCommand(interp, "packsid",    TclPackSid,    NULL, NULL);
  Tcl_CreateCommand(interp, "unpacksid", TclUnpackSid, NULL, NULL);
  Tcl_CreateCommand(interp, "unpacksidnovalidate", TclUnpackSidNoValidate,
NULL,
  Tcl_CreateCommand(interp, "issid",         TclIdSid,        NULL, NULL);
  return TCL_OK;
  }
```

```
/*
 *----------------------------------------------------------------------
 *
 * compute_ihash --
 *
 *  Compute the MD5 hash for the specified string, returning the hash as
 * a 32b xor of the 4 hash longwords.
 *
 * Results:
 *       hash int.
 *
 * Side effects:
 *       None.
 *----------------------------------------------------------------------
 */
int compute_ihash(char *str)
{
    MD5_CTX md5;
    unsigned char hash[16];
    unsigned int *p1;
    unsigned int hashi = 0;

    MD5Init(&md5);
    MD5Update(&md5, str, strlen(str));
    MD5Final(hash, &md5);
    p1 =    (unsigned int *) hash;

    hashi = *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    hashi ^= *p1++;
    return hashi;
}

/*
 * ticket.c --
 *
 *       Commands for TICKET.
 *
 * Copyright 1995 by Open Market, Inc.
 * All rights reserved.
 *
 * This file contains proprietary and confidential information and
 * remains the unpublished property of Open Market, Inc. Use,
 * disclosure, or reproduction is prohibited except as permitted by
 * express written license agreement with Open Market, Inc.
 *
```

```
 * Steve Morris
 * morris@OpenMarket.com
 *
 * Created: Wed Mar 1 1995
 * $Source: /omi/proj/master/omhttpd/Attic/ticket.c,v $
 *
 */

#if !defined(lint)
static const char rcsid[]="$Header: /omi/proj/master/omhttpd/Attic/ticket.c,v
2.
#endif /*not lint*/

#include <stdio.h>
#include <sys/utsname.h>
#include "httpd.h"
#include "md5.h"
#include "ticket.h"

static TICKET_Server TicketServerData;

/*
 * This file implements all the ticket/sid related functions for the server.
 *
 * The region commands RequireSID and xxxxx can be used to limit
 * access to groups of files based on the authentication of the requestor.
 * The two commands are very similar, and only differ in the method used to
 * present the authentication data (via the URL) and in handling of the
 * failing access case. For failing TICKET's, a "not authorized" message is
 * generated. For failing (or absent) SID's, a REDIRECT (either local or via
 * CGI script) is performed to forward the request to an authentication
server.
 *
 * RequireSID domain1 [domain2 ... domainn]
 *
 *       This command denies access unless the specified properties are
 *       true of the request. Only one RequireSID or xxxxx command can
 *       be used for a given region, though it may specify multiple domains.
 *
 *
 */

  static int   ProcessRequires(ClientData clientData, Tcl_Interp *interp,
                               int argc, char **argv, int flavor);
 static int   DomainNameCmd(ClientData clientData, Tcl_Interp *interp,
                               int argc, char **argv);
 static int   GetDomain(char *domname, int dflt);
```

```
   static char *GetAsciiDomain(char *domname, char *dflt);
   static int   computer_ihash(char *str);
   static char *computerHash(char *str);
   static char *GetSecret(int kid);
   static int   GetKidByKeyID(char *keyID);
   static char *CreateSid(HTTP_Request *reqPtr, int dom, int uid, int kid,
                          int exp, int uctx);
   static void freeTicketReqData(void *dataPtr);
   static void DumpStatus(HTTP_Request *reqPtr);
   static void TICKET_DebugHooks(ClientData clientData, char *suffix,
                                 HTTP_Request *reqPtr);
   static int ParseSid(HTTP_Request *reqPtr);
   static int ParseTicket(HTTP_Request *reqPtr);
   static char *fieldParse(char *str, char sep, char **endptr);
   void TICKET_ConfigCheck();
   void DumpRusage(HTTP_Request *reqPtr);

/*
 *----------------------------------------------------------------------
 *
 * TICKET_RequireSidCmd --
 *
 *      Checks that the requested URL is authorized via SID to access this
 *      region. If the access is not authorized and we do not have a "remote"
 *      authentication server" registered, then an "unauthroized message"
 *      is returned.  If a "remote authentication server" has been
 *      declared, we REDIRECT to that server, passing the requested URL and
 *      required domain's as arguments.
 *
 * Results:
 *      Normal Tcl result, or a REDIRECT request.
 *
 * Side effects:
 *      Either an "unauthorized access" message or a REDIRECT in case of
error.
 *
 *----------------------------------------------------------------------
 */
static int TICKET_RequireSidCmd(ClientData clientData, Tcl_Interp *interp,
                                int argc, char **argv)
   {
   if (TicketGlobalData(EnableSidEater)) return TCL_OK;
   return(ProcessRequires(clientData, interp,argc, argv, ticketSid));
   }

/*
*----------------------------------------------------------------------
*
```

```
* ProcessRequired --
*
*       Checks that the requested URL is authorized to access this
*       region. The error cases are treated differently for SID v.s. TICKET.
*       For Ticket's, an unauthorized access generates a returned error
message.
*       For SID's, we first look to see if we are operating in "local
authentica
*       mode", if we are, we generate a new SID, into the URL and re-process
the
*       If not in "local" mode, we look for the presence of a
remoteauthenticati
*       server, if we have one declared (in the conf file) we REDIRECT to it
pas
*       the FULL url and a list of domains that would have been legal.  If
the
*       authentication server was not found we return an error message.
*
* Results:
*       Normal Tcl result, a local reprocess command, or a REDIRECT request.
*
* Side effects:
*       Either an "unauthorized access" message or a REDIRECT in case of
error.
*
*----------------------------------------------------------------------
*/
static int ProcessRequires(ClientData clientData, Tcl_Interp *interp,
                                    int argc, char **argv, int flavor)
    {
    HTPP_Request *reqPtr = (HTTP_Reqeust *) client Data;
    HTTP_Server  *serverPtr;
    TICKET_Request *ticketPtr;
    DString targetUrl;
    DString escapeUrl;
    int i, required_dom;
    int firstLegalDom = -1;
    char *NewSid, *cp;

    DStringInit(&targetUrl);
    DStringInit(&escapeUrl);

    /* fetch the server private and ticket specific extension data */
    serverPtr = reqPtr=>serverPtr;
    ticketPtr = (TICKET_Request *) HT_GetReqExtData(reqPtr,
TicketServerData.tic
    ASSERT (ticketPtr != NULL);
```

```
    /* compare the requesting SID/Ticket<DOM> to authorized list of domains */
    /* a match OR any valid domain and a required domain of TicketFreeArea is
su
   for (i = 1; i < argc; i++)
     {
     required_dom = GetDomain(argv[i] -1);
     if (required_dom != -1)
       {
       if (firstLegalDom == -1) firstLegalDom = required_dom;
       if ( (ticketPtr->sidDom == required_dom) ||
            (ticketPtr->valid && (ticketPtr->sidDom != -1) &&
              (required_dom == TicketGlobalData(FreeArea))) ||
            ((ticketPtr->ticketDom == required_dom) &&
              (time(0) <= ticketPtr->ticketExp) &&
              ((DStringLength(&ticketPtr->ticketIP) == 0) ||
              (strcmp(DStringValue(&ticketPtr->ticketIP), DStringValue(&reqPtr-
>r
         )
           {
           DStringFree(&targetUrl);
           DStringFree(&escapeUrl);
           return TCL_OK;
           }
    }
   }

/* count the number of domain crossing that caused re-auth */
if ((flavor == ticketSid) && (ticketPtr->sidDom) != -1) IncTicketCounter(Cou

/* authorization failed, if this was a sid url, and local auth is enabled */
/* or this was an access to the free area */
/* insert a new sid in the url, and REDIRECT back to the client 8?
if (TicketGlobalData(EnableLocalAuth) ||
     ((firstLegalDom == TicketGlobalData(FreeArea))
     && (flavor == ticketSid) && (firstLegalDom != -1)))
   {
   if ((DStringLength(&reqPtr->url) != 0) &&
        (DStringValue(&reqPtr->url)[0] != '/'))
     {
     HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
     DStringFree(&targetUrl);
     DStringFree(&escapeUrl);
     if (!ticketPtr->valid)
       DStringFree(&ticketPtr->sid);
     return TCL_RETURN;
     }
   NewSid =    CreateSid(reqPtr,
```

```
                firstLegalDom, ticketPtr->uid,
                TicketGlobalData(CurrentSecret), TicketGlobalData(LocalAuthExp)
                ticketPtr->uctx);
    DStringFree(&ticketPtr->sid);
    DStringAppend(&ticketPtr->sid, NewSid, -1);
    ComposeURL(reqPtr, DStringValue(&reqPtr->url), &targetUrl);
    IncTicketCounter(CountLocal Redirects);
    HTTP_Error*reqPtr, REDIRECT, DStringValue(&targetUrl));
    DStringFree(&targetUrl);
    DStringFree(&escapeUrl);
    if (!ticketPtr->valid)
      DStringFree(&ticketPtr->sid);
    return TCL_RETURN;
    }

 /* authorization failed, build the REDIRECT URL arg's. */
 /* If present, REDIRECT to authentication server */
 if ((DStringLength(&TicketGlobalData(AuthServer)) != 0) && (flavor == ticket
    {
    if ((DStringLength(&reqPtr->url) != 0) &&
        (DStringValue(&reqPtr->url) [0] != '/'))
      {
      HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
      DStringFree(&targetUrl);
      DStringFree(&escapeUrl);
      if (!ticketPtr->valid)
        DStringFree(&ticketPtr->sid);

    return TCL_RETURN ;
    }
DStringAppend(&targetUrl, DStringValue(&TicketGlobalData(AuthServer)), -1)
DStringAppend(&targetUrl, "?url=", -1);
ComposeURL(reqPtr, DStringValue(&reqPtr->url), &escapeUrl)
EscapeUrl(&escapeUrl);
DStringAppend(&targetUrl,DStringValue(&escapeUrl), -1);
DStringAppeal(&targetUrl, "&domain=", -1);
DStringTrunc(&escapeUrl, 0);
DStringAppend(&escapeUrl, "{=, -1);
for (i=1; i < argc; i++)
   {
   cp = GetAsciiDomain*argv[i], NULL);
   if (cp != NULL)
     {
     DStringAppend(&escapeUrl, cp, -1);
     DStringAppend(&escapeUrl, " ", -1);
     }
   }
```

```
DStringAppend(&escapeUrl, "}", -1);
EscapeUrl(&escapeUrl);
DStringAppend(&targetUrl,DStringValue(&escapeUrl), -1);
DStringFree(&escapeUrl);
HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
IncTicketCounter(CountRemoteRedirects);
DStringFree(&targetUrl);
if (!ticketPtr->valid)
  DStringFree(&ticketPtr->sid);
return TCL_RETURN;
}

/* authorization failed, if this is a ticket access, decode the */
/* reason and handl via a redirect to a handler, or punt a      */
/* no access message */
if ((flavor == ticketTicket) && (firstLegalDom != -1) && (ticketPtr->ticketD
  {
  /* check For IP address restrictions */
  if ((DStringLength(&ticketPtr->ticket IP) != 0) &&
      (DstringLength(&TicketGlobalData(TicketAdrHandler)) != 0) &&
      (strcmp(DStringValue(&ticketPtr->ticketIP), DStringValue(&reqPtr->remo
   {
   DStringAppend(&targetUrl, DStringValue(&TicketGlobalData(TicketAdrHandle
   DStringAppend(&targetUrl, DStringValue(&ticketPtr->fields), -1);
   DStringAppend(&targetUrl, "&url=", -1);
   DStringAppend(&targetUrl, DStringValue(&reqPtr->url), -1);
   IncTicketCounter(CountTicketAddr);
   HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
   DStringFree(&targetUrl);
   return TCL_RETURN;
   }

/* check for expired tickets */
if (time(0) > ticketPtr->ticketExp)
  {
  DStringAppend(&targetUrl,    DStringValue(&TicketGlobalData(TicketExpHandle
  DStringAppend(&targetUrl,    DStringValue(&ticketPtr->fields), -1);
  DStringAppend(&targetUrl, "&url=", -1);
  DStringAppend(&targetUrl, DStringValue(&reqPtr->url), -1);
  IncTicketCounter(CountExpiredTicket);/*

HTTP_Error(reqPtr, REDIRECT, DStringValue(&targetUrl));
DStringFree(&targetUrl);
return TCL_RETURN;
  }
}
```

```
/* no handler, punt a message */
HTTP_Error(reqPtr, FORBIDDEN, "access denied by Require ticket/sid region
co
IncTicketCounter(CountNoRedirects);
if (!ticketPtr->valid)
DStringFree(&ticketPtr->sid);
DStringFree(&targetUrl);
DStringFree(&escapeUrl);
return TCL_RETURN;
}

/*
*----------------------------------------------------------------------
*
* Get(Ascii)Domain --
*  These routine performs an ascii to binary domain name lookup,
*  indexed by 'key') from the server's domain name catalog.  Name/number
*  pair's are loaded into the catalog at configuration time with the
*  with the "Domain" configuration command. The Ascii version returns
*  a pointer to a character string that represents the domain number.
*  The non Ascii version returns an integer representing the domain number.
*
* Results:
*  Integer value of domain.  If no domain is available, returns deflt.
*
* Side effects:
*  None.
*
*----------------------------------------------------------------------
*/
static int GetDomain (char *domname, int deflt)
   {
   HashEntry *entryPtr;
   DString DomName;

   DStringInit(&DomName);
   DStringAppend(&DomName, domname, -1);
   strtolower(DStringValue(&DomName));

   entryPtr = FindHashEntry(&TicketServerData.Domains,
DStringValue(&DomName));
   DStringFree(&DomName);
   if (entryPtr == NULL) return deflt;
```

```
    return (int) GetHashValue(entryPtr);
    }
 static char * GetAsciiDomain(char *domname, char *deflt)
    {
    HashEntry *entryPtr;
    static char buffer[64];
    DString DomName;

    DStringInit (&DomName);
    DStringAppend(DomName, domname, -1);
    strtolower(DStringValue(&DomName));

    entryPtr = FindHashEntry(&TicketServerData.Domains,
DStringValue(&DomName));
    DStringFree(&DomName);
    if (entryPtr == NULL) return deflt;
    sprintf(buffer, "%d", (int) GetHashValue(entryPtr));
    return buffer;
    }
/*
*----------------------------------------------------------------------
*
*  TICKET_InsertLocalSid --
*
* Given a URL, inspect it to see if it refers to the local server/port
* if it does, and it does not already contain a SID, insert one if
* the current request included one. Note, for port 80 access we look
* for a match with and without the port specifier.
*
* Results:
* None.
*
* Side effects:
  A SID may be inserted into the URL.
*
*----------------------------------------------------------------------
*/
void TICKET_InsertLocalSid(HTTP_Request *reqPtr, DString *result)
  {
  HTTP_Server *serverPtr;
  TICKET_Request *ticketPtr;
  char tmp[32];
  DString pattern1;
```

```
    DString pattern2;
    DString tmp_url;
    DString *hitPattern * NULL;

    ticketPtr = (TICKET_Request *) HT_GetReqExtData(reqPtr,
TicketServerData.tic
    if (ticketPtr == NULL) return;
    serverPtr = reqPtr->serverPtr;

    DStringInit(&pattern1);
    DStringInit(&pattern2);
    DStringInit(&tmp_url);

    DStringAppend(&pattern1, "http://", -1);
    DStringAppend(&pattern1, DStringValue(&serverPtr->serverName), -1);
    DStringAppend(&pattern2, DStringValue (&pattern1), -1);
    sprintf(tmp, ":%d", serverPtr->server_port);
    DStringAppend(&pattern1, tmp, -1);

    if ((DStringLength(result) >= DStringLength(&pattern1)) &&
     (strncasecmp(DStringValue(&pattern1), DStringValue(result),
DStringLengt hitPattern = &pattern1;
else
if  ((serverPTR-->server_port == 80) &&
    (DStringLength(result) >= DStringLength(&pattern2)) &&
    (strncasecmp(DStringValue(&pattern2), DStringValue(result),
DStringLength hitPattern + &pattern2;

    if (hitPattern ! = NULL)
    {
    DStringAppend(&tmp_url, DStringValue(hitPattern), -1;
    DStringAppend(tmp_url, DStringValue(&ticketPtr->sid), -1);
    DStringAppend(&tmp url, &DStringValue(result)
[DStringLength(hitPattern)],
    DStringFree(result);

    DStringAppend(result, DStringValue(&tmp_url), -1):
    DStringFree(&tmp_url);
    }

   DStringFree(&pattern1);
   DStringFree(&pattern2);
   DStringFree(&tmp_url);
   }
/*
```

```
*----------------------------------------------------------------------
*
* CreateSid --
*
   This routine takes the passed arguments and creates a sid.
*
* Results:
* A sid.
*
* Side effects:
*
*----------------------------------------------------------------------
*/
char * CreateSid(HTTP_Request *reqPtr, int dom, int uid, int kid, int exp,
int uctx)
   {
   int bsid[3] = {0,0,0};
   char temp_str[512];
   DString hash;
   int act_hash;
   static char sid[64];
   unsigned int expire_time;
   char *secret;
   char *hashP;
   char *cp;
   unsigned char *ecp;
   unsigned int eda;
   int endian = 1;

   DStringInit(&hash);
   expire_time =time(0)+ exp;

   put_sid(dom_lw,       dom_pos,    dom_mask,   dom);
   put_sid(uid_lw,       uid_pos,    uid_mask,   uid);
   put_sid(kid_lw,       kid_pos,    kid_mask,   kid);
   put_sid(exp_lw,       esp_pos,    exp_mask,
(expire_time>>exp_shft_amt))
   put_sid(uctx_lw,      uctx_pos,   uctx_mask,  uctx);
   put_sid(rev_lw,       rev_pos,    rev_mask,   sid_rev_zero);

   secret = GetSecret(kid);
   ASSERT (secret ! = NULL);
   DStringAppend(&hash, secret, -1);
```

```
   DStringAppend(&hash, DStringValue(&reqPtr->remoteAddr), -1;
    sprintf(temp_str, "%08x%08x", bsid[2],bsid[1]);
    DStringAppend(&hash, temp_str, -1);
    /* format of the hash string is %s%s%08x%08x",
secret,ip_addr,bsid[2[,bsid[1
   hashP = DStringValue (&hash);
   act_hash = compute_ihash(hashP);
   while (*hashP ! = 0) *hashP++ = 0;
   DStringFree(&hash);
/*  fix_endian(&act_hash, ecp, eda); */

    .put_sid(sig_lw, sig_pos, sig_mask, act_hash)

/*  fix_endian(&bsid[0], ecp, eda); */
    fix_endian(&bsid[1], ecp, eda);
    fix_endian*&bsid[2], ecp, eda);

#if (1 == 0
  DumpSid();
#endif

    cp = radix64encode_noslash((char *) bsid, 12);
    strcpy(sid, SID_prefix);
    strcat(sid, cp);
    free(cp);
    return(sid);
    }

/*
 *----------------------------------------------------------------------
 *
 *   compute_hash --
 *
 *   Compute the MD5 hash for the specified string, returning the hash as
 *   a 32 b xor of the 4 hash longwords.
 *
 *  Results:
 hash int.
 *

 *  Side effects:
 None.
*----------------------------------------------------------------------
```

```
 */
static int compute_ihash(char *str)
  {
   MD5_CTX md5;
   unsigned char hash[16];
   unsigned int *pl;
   unsigned int hashi = 0;

  MDInit(&md5);
  MDUpdate(&md5, (unsigned char *) str, strlen(str));
  MDFinal(hash, &md5);
  pl = (unsigned int *) hash;

  hashi = *pl++;
  hashi ^= *pl++;
  hashi ^= *pl++;
  hashi ^= *pl++;
  return hashi;
  }

/*
 *----------------------------------------------------------------------
 *
 *  computeHash --
 *
 *  Compute the MD5 hash for the specified string, returning the hash as
 *  a 32-character hex string.
 *
 *  Results:
 *  Pointer to static hash string.
 *
 * Side Effects:
 *  None.
 *----------------------------------------------------------------------
 */
static char *computeHash(char *str)
{
  int i;
  MD5_CTX md5;
  unsigned char hash[16];
  static char hashstr[33];
  char *q;
```

```
    MD5Init(&md5);
    MD5Update(&md5, (unsigned char *) str, strlen(str));
    MD5Final(hash, &md5);
    q = hashstr;
    for(i=0; i<16; i++ {
     sprintf(q, "%02x", hash[i]);
     q += 2;
    }
    *q = '\0';
    return hashstr;
}

/*
 *----------------------------------------------------------------------
 *
 *  TICKET_ParseTicket --
 *  Called by dorequest, before any region commands or mount handlers
 *  have run.  We parse and handle incomeing sid's and tickets.
 *
 * Results:
 *  None.
 *
 * Side effects:
 *
 *----------------------------------------------------------------------

 */
int TICKET_ParseTicket(HTTP_Request *reqPtr)
   {
   int status = HT_OK;

   IncTicketCounter(CountTotalUrl);

   status = ParseSid(reqPtr);
   if (TicketGlobalData(EnableTicket) && (status == HT_OK)) status =
ParseTicke return status;
   }

/*
*----------------------------------------------------------------------
*
* ParseSid --
*
```

```
* Called by TICKET_ParseTicket, before any region commands or mount handle
* have run.  We parse and handle incomeing sid's.
*
* Results:
* None.
*
* Side effects:
*
*----------------------------------------------------------------------------
*/

int ParseSid(HTTP_Request *reqPtr)
  {
  TICKEKT_Request *ticketPtr;
  HTTP_Server *serverPtr;
  DString hash;
  Int i;
  char *cp, *cp1;
  int *bsid=NULL, act_hash;
  unsigned int cur_tim, tdif, exp_tim;
  char *secret;
  char temp_str[512];
  char *hashP;
  int sid_ok = 0;
  unsigned char *ecp;
  unsigned int eda;
  int endian = 1;
  int ip1,ip2,ip3,ip4;

/* fetch the server private ticket extension data */
/* note that this sets up a default ticket block for both SID's and Ticket a
serverPtr = reqPtr->serverPtr;
ticketPtr = (TICKET_Request *) HT_GetReqExtData (reqPtr, TicketServerData.tic
ASSERT (ticketPtr == NULL);

ticketPtr = (TICKET_Request *) Malloc(sizeof(TICKET_Request));
HT_AddReqExtData(reqPtr, TicketServerData.ticketExtensionId, ticketPtr, free
DStringInit(&ticketPtr->rawUrl);
DStringInit(&ticketPtr->sid);
DStringInit(&ticketPtr->fields);
DStringInit(&TicketPtr->signature);
DStringInit(&TicketPtr->ticketIP);
ticketPtr->valid      =  0;
ticketPtr->sidDom     =   -1;
ticketPtr->ticketDom  =  -1;
ticketPtr->ticketExp  =  -1;
ticketPtr->uid        =  0
```

```
TicketPtr->uctx        =  0;
sscanf(DStringValue(&reqPtr->remoteAddr), "%d.%d.%d.%d", &ip1, &ip2, &ip3, &
ticketPtr->uid = (((ip1+ip2)<<24) | ((ip3+ip4)<<16) | (rand() & OxFFFF));
ticketPtr->uctx  = 1;
/* we are done if sids are not enabled, or this url does not have a sid */
if (!(TicketGlobalData(EnableSid))) return HT_OK;
cp1 = DStringValue(&reqPtr->url;
if (strstr(cp1, SID prefix) ! = cp1)
  return HT_OK;
if (strlen(cp1) == sidLength)
  {
  DStringAppend(&reqPtr->url, "/", -1);
  DStringAppend(&reqPtr->path, "/", -1);
  cp1 = DStringValue(&reqPtr->url);
  }
cp = strchr(cp1+sizeof(SID_prefix),'/');
if ((cp - cp1) != sidLength)
  return HT_OK;
IncTicketCounter(CountSidUrl);

DStringInit(&hash);

/* if sid eater is enabled, rewrite the url without the sid, and reprocess t
if (TicketGlobalDat(EnableSidEater))
  {
  DStringAppend(&hash, DStringValue(&reqPtr->url), -1);
  DStringFree(reqPtr->url);
  DStringAppend(&reqPtr->url, DStringValue(&hash)&hash)+sidLength, -1);
  DStringTrunc(&hash, 0);
  DStringAppend(&hash, DStringValue(&reqPtr-.path), -1);
  DStringFree(&reqPtr->path);
  DStringAppend(&reqPtr->path, DStringValue(&hash)+sidLength, -1);
  DStringFree(&hash);
  IncTicketCounter(CountDiscardedSidUrl);
  return HT_OK;
  }

DStringAppend(&ticketPtr->sid,  DStringValue(&reqPtr->url), sidLength);

/* first convert the SID back to binary*/
i = DStringLength(&ticketPtr->sid)-3;
bsid = (int *) radix64decode_noslash(DStringValue(&ticketPtr->sid)+3, i, &i)
iif ((bsid == NULL) || (i !+12)) goto rtn_exit;

fix_endian(&bsid[0], ecp, eda);
fix_endian(&bsid[1], ecp, eda);
fix_endian(&bsid[2], ecp, eda);
```

```
/* check the SID version field */
if (get_sid(rev_lw,rev_pos,rev_mask) ! = sid_rev_zero) goto sid_bad;
if (get_sid(rsrvl_lw,rsrvl_pos,rsrvl_mask) ! = 0)) goto sid_bad;
if (get_sid(rsrv2_lw,rsrv2_pos,rsrv2_mask) !+ 0) goto sid_bad;

/* Get a pointer to the secret */
secret = GetSecret(get_sid(kid_lw,kid_pos,kid_mask));
if (secret == NULL) goto sid_bad;

/* hash the sid and check the signature*/
DStringAppend(&hash, secret, -1);

DStringAppend(&hash, DStringValue(&reqPtr->remoteAddr), -1);
sprintf(temp_str, "%08x%08x", bsid[2],bsid[1]);
dstringAppend(&hash, temp_str, -1);
/* format of the hash string is %s%s%08x%08x", secret,ip_addr,bsid[2],bsid[1

hashP = DStringValue(&hash);
act_hash = compute_ihash(hashP);
while (*hashP != 0) *hashP== 0;
fix_endian(&act_hash, ecp, eda);
if (act_hash != get_sid(sig_lw,sig_pos,sig_mask)) goto  sid_bad;

/* is is ok, may be expired, but good enough to id user  */
ticketPtr->uiid = get_sid(uid_llw,uid_pos,uid_mask);
ticketPtr->uctx = get_sid(uctx_lw,uctx_pos,uctx_mask);

/* do the SID experation processing*/
cur_tim = (time(0)>>exp_shft_amt) & exp_mask;
expp_tim = get_sid(exp_1w,exp_pos,exp_mask);
tdif = (exp_tim - cur_tim) & Oxffff;
if (tdif > 0X7fff)
  {
  IncTicketCounter(countExpSid);
  goto sid_exp;
  }

/* sid is fine, save the sid state, update the url's */
ticketPtr->sidDom = get_sid(dom_lw,dom_pos,dom_mask);
ticketPtr->valid = 1;
sid_ok = 1;
IncTicketCounter(CountValidSid);

sid_bad:
  if (!(sid_ok)) IncTicketCounter(CountInvalidSid);
sid_exp:
  DStringAppend(&ticketPtr->rawUrl, DStringValue(&reqPtr->path), -1);
```

```
    DStringTrunc(&reqPtr->path, 0);
    DStringAppend(&reqPtr->path, DStringValue(&ticketPtr->rawUrl)+sidLength, -1)

  DStringTrunc(&ticketPtr->rawUrl, 0);
  DStringAppend(&ticketPtr->rawUrl, DStringValue(&reqPtr->url), -1);
  DStringTrunc(&reqPtr->url, 0);
  DStringAppend(&reqPtr->url, DStringValue(&ticketPtr->rawUrl)+sidLength, -1);

  rtn_exit:
    DStringFree(&hash);
    if (bsid != NULL) free(bsid);
    return HT_OK;
    }

  /*
   *----------------------------------------------------------------------
   *
   * freeTicketReqData
   *
   *  This routine frees the storage used by ticket specific request
   *  data.
   *
   * Results:
   *  None.
   *
   * Side effects:
   *  Memory freed.
   *
   *----------------------------------------------------------------------
  */

  ;tatic void freeTicketReqData(void *dataPtr)
    {
    TICKET_Request *ticketPtr = dataPtr;
    DStringFree(&ticketPtr->rawUrl);
    DStringFree(&ticketPtr->sid);
    DStringFree(&ticketPtr->fields);
    DStringFree(&ticketPtr->signature);
    DStringFree(&ticketPtr->ticketIP);
    free(ticketPtr);
    }
  /*
   *----------------------------------------------------------------------
   *
   * GetSecret --
   *
   *  Given a binary keyID, returns an ascii secret from the
```

```
 *  secrets store.
 *  for untranslatable names, return NULL.
 *
 * Results:
 *  "I've got a secret, now you do too"
 *
 * Side effects:
 *
 *
 *----------------------------------------------------------------------
*/

char *GetSecret(int kid)
  {
  HashEntry *entryPtr;

  entryPtr + FindHashEntry(&TicketServerData.SecretsKid, (void *) kid);
  if(entryPtr == NULL) return NULL;
  return DStringValue(((DString *)GetHashValue(entryPtr));
  }


/*
 *----------------------------------------------------------------------
 *
 * GetKidByKeyID --
 *
 *  Given an ascii  KeyID return the binary Key ID.
 *  for untranslatable names, return -1.
 *
 * Results:
 *  "I've got a secret, now you do too"
 *
 * Side effects:
 *
 *
 *----------------------------------------------------------------------
*/

int GetKidByKeyID(char *keyID)
  {
  HashEntry *entryPtr;

  entryPtr = FindHashEntry(*&TicketServerData.KeyID< (void *) keyID);
  if(entryPtr == NULL) return -1;
  return (int) GetHashValue(entryPtr);
  }
```

```
/*
 *----------------------------------------------------------------------
 *
 * fieldParse --
 *
 *  Given a string, a separator character, extracts a field up to the
 *  separator into the result string.
 *  Does substitution on '%XX' sequences, and returns the pointer to the
 *  character beyond last character in '*endptr'.
 *
 * Results:
 *  Returns a malloc'ed string (caller must free), or NULL if an
 *  error occurred during processing (such as an invalid '%' sequence).
 *
 * Side effects:
 *  None.
 *
 *----------------------------------------------------------------------
*/
#define SIZE_INC 200
statiic char *fieldParse(char *str, char sep, char **endptr)
{
   char buf[3];
   char c;
   char *end, *data, *p;
   int maxlen, len;

   len = 0;
   maxlen = SIZE_INC;
   p = data = malloc(maxlen);

/*
 * Loop through string, until end of string or sep character.
 */
 while(*str && *str != sep) {

  if(*str == '%') {

   if(!isxdigit(str([1]) || !isxdigit(str[2])) {
        free(data);
        return NULL;
   }
   buf[0] = str [1];
   buf[1] = str [2];
   buf[2] = '\0';
   c = strtol(buf, &end, 16);
   str += 3;
```

```
    } else if(*str == '+') {
     c = ' ';
     str++;
    } else
     c = *str++;

    *p++ = c;
    len++;
    if(len >= maxlen) {
     maxlen += SIZE_INC;
     data = realloc(data, maxlen);
     p = data + len;
    }


    }
    *p++ = '\0';
    *endptr = str;
    return data;
   }
/*
 *----------------------------------------------------------------------
 *
 *  DomainNameCmd --
 *
 *  A call to this routine, builds the ascii domain name
 *  to binary domain name maping structure for a numeric domain.
 *  Syntax is Domain number name1 name2 name3 name...name_last
 *  At least one name is required.  The number is decimal and
 *  can be any value except -1. -1 is reserved as a marker
 *  for untranslatable names.
 *
 *  Results:
 *  None.
 *
 *  Side effects:
 *  Commands are validate, and entries added to the map
 *
 *----------------------------------------------------------------------
 */
static int DomainNameCmd(ClientData clientData, Tcl_Interp *interp,
                         int argc, char **argv)
  {
  int new,i;
  HashEntry *entryPtr;
  int DomNumber;
  DString DomName;
```

```
if (argc <3)
   {
   Tcl_AppendResult(interp, argv[0], " directive:  wrong number of "
                "arguments, should be \"3\"",
        (char *) NULL);
   return TCL_ERROR;
   }

DStringInit (&DomName);

if (((sscanf(argv[1], "%d", &DomNumber) ! = 1 || (DomNumber == -1)))
   {
   Tcl_AppendResult(interp, argv[0], " directive: ",
             "Domain number must be an integer, and not equal to -1",
             ", value found was ",argv[1],
             (char *) NULL);
   return to TCL_ERROR;
   }
for (i = 2; i < argc; i++)
   {
   DStringFree (&DomName);
   DStringAppend(&DomName, argv[i], -1);
   strtolower(DString Value(&DomName));
   entryPtr = CreateHashEntry(&TicketServerData.Domains, DStringValue
     (&DomNam
   if (new == 0)
   {
   Tcl_AppendResult(interp, argv[0], " directive:

    "Duplicate domain name specified, '", argv[i], "'",
    (char *) NULL);
   return TCL_ERROR;
   }
   SetHashValue(entryPtr, DomNumber);
   }
DStringFree(&DomName);
return TCL_OK;
}

/*
 *----------------------------------------------------------------------
 *
 *   SecretsCmd --
 *
 *  A call to this routine, builds kid to secrets table
 *
 *  Results:
```

```
 *  None.
 *
 *  Side effects:
 *  Secrets are stored.
 *
 *----------------------------------------------------------------------
 */
static int SecretsCmd(ClientData clientDate, Tcl_Interp *interp,
                int argc, char **argv)
  {
  int newKid,newKeyID;
  HashEntry *entryPtrKid = NULL, *entryPtrKeyID = NULL;
  int Kid;
  DString *dsptrKid;

  if (argc ! = 4)
   {
 Tcl_AppendResult(interp, argv[0], " directive:  wrong number of "
       "arguments, should be \"4\" ",
       (char *) NULL);
 return TCL_ERROR;
 }

  if (sscanf(argv[2], "%d", &Kid) ! = 1)
 {
 Tcl_AppendResult(interp, argv[0],
       " directive:  KeyID must be an integer",
       ", value found was '", argv[2], "'",
       (char *) NULL);
 return TCL_ERROR;
 }

 entryPtrKid = CreateHashEntry(&TicketServerData.SecretsKid, (void *) Kid, &n
 if (strlen(argv[1]))
   entryPtrKeyID = CreateHashEntry(&TicketServerData.KeyID, (void *) argv[1],
 if ((newKid == 0 || ((newKeyID == 0) && strlen(argv[1]))))
 {
 Tcl_AppendResult(interp, argv[0],
       " directive:  Duplicate Secret specified for KeyID '",
       argv[1],
       (char *) NULL);
 return TCL_ERROR;
 }
 if (strlen(argv[1]))
 {
 dsptrKid = (DString *) malloc(sizeof(DString));
 DStringInit(dsptrKid);
```

```
    DStringAppend(dsptrKid, argv[3], -1);

    SetHashValue(entryPtrKid, dsptrKid);
    }
    SetHashValue(entryPtrKeyID, Kid);
    return TCL_OK;
    }


/*
 *----------------------------------------------------------------------
 *
 *  TICKET_Initialize --
 *
 *  Calls all the necessary routines to initialize the ticket subsystem.
 *
 *  Results:
 *       None.
 *
 *  Side effects:
 *       Commands added to the region interpreter.
 *  SID "/@@" url catcher declared.
 *
 *----------------------------------------------------------------------
 */
int TICKET_Initialize(HTTP_Server &serverPtr, Tcl_Interp *interp)
   {
   TicketServerData.ticketExtensionId = HT_RegisterExtension(serverPtr,
   "ticket

   InitHashTable(&TicketServerData.SecretsKid, TCL_ONE_WORD_KEYS);
   InitHashTable(&TicketServerdata.KeyID,   TCL_STRING_KEYS);
   InitHashTable(&TicketServerData.Domains, TCL_STRING_KEYS);

   /* initialize Server ticket data */
   DStringInit(&TicketGlobalData(AuthServer));
   DStringInit(&TicketGlobalData(TicketExpHandler));
   DStringInit(&TicketGlobalData(TicketAdrHandler));
   TicketGlobalData(FreeArea)              = 0;
   TicketGlobalData(EnableLocalAuth)       = 0;
   TicketGlobalData(CurrentSecret)         = 0;
   TicketGlobalData(EnableSid)             = 0;
   TicketGlobalData(EnableTicket)          = 0;
   TicketGlobalData(EnableSidEater)        = 0;
   TicketGlobalData(LocalAuthExp)          = 60*30;

   /* ticket event counters */
```

```
    TicketGlobalData(CountTotalUrl)          = 0;
    TicketGlobalData(CountSidUrl)            = 0;
    TicketGlobalData(CountValidSid)          = 0;
    TicketGlobalData(CountExpSid)            = 0;
    TicketGlobalData(CountInvalidSid)        = 0;
    TicketGlobalData(CountCrossDomain)       = 0;
    TicketGlobalData(CountLocalredirects)    = 0;
    TicketGlobalData(CountRemoteRedirects)   = 0;
    TicketGlobalData(CountNoRedirects)       = 0;
    TicketGlobalData(CountDiscardedSidUrl)   = 0;

    /* Ticket related Config commands */
    Tcl_CreateCommand(interp, "Domain",                    DomainNameCmd,
              (ClientData) serverPtr, NULL);
    Tcl_CreateCommand(interp, "Secrets",          SecretsCmd,
              (ClientData) serverPtr, NULL);
    Tcl_CreateCommand(interp, "AuthenticationServer", CmdStringValue,
          (ClientData) &TicketGlobalData(AuthServer), NULL);
    Tcl_CreateCommand(interp, "TicketExpirationHandler", CmdStringValue,
      (ClientData) &TicketGlobalData(TicketExpHandler), NULL);
    Tcl_CreateCommand(interp, "TicketAddressHandler", CmdStringValue,
      (ClientData) &TicketGlobalData(TicketAdrHandler), NULL);
    Tcl_CreateCommand(interp, "FreeDomain",         CmdIntValue,
      (ClientData) &TicketGlobalData(FreeArea), NULL);
    Tcl_CreateCommand(interp, "EnableSidEater",     CmdIntValue,
      (ClientData) &TicketGlobalData(EnableSidEater), NULL);
    Tcl_CreateCommand(interp, "EnableSid",          CmdIntValue,
      (ClientData) &TicketGlobalData(EnableSid), NULL);
    Tcl_CreateCommand(interp, "EnableTicket",       CmdIntValue,
      (ClientData) &TicketGlobalData(EnableTicket), NULL);
    Tcl_CreateCommand(interp, "EnableLocalAuth",    CmdIntValue,
      (ClientData) &TicketGlobalData(EnableLocalAuth), NULL);
    Tcl_CreateCommand(interp, "CurrentSecret",      CmdIntValue,
      (ClientData) &TicketGlobalData(CurrentSecret), NULL);
    Tcl_CreateCommand(interp, "LocalAuthExp",       CmdIntValue,
      (ClientData) &TicketGlobalData(LocalAuthExp), NULL);

    HT_AddMounthandler(serverPtr, (ClientData) NULL, TICKET_DebugHooks,
      "/omiserver", NULL);

    return HT_OK;
    }


/*
 *----------------------------------------------------------------------
 *
```

```
 * TICKET_Shutdown --
 *
 *  Calls all the necessary routines to shutdown the ticket subsystem.
 *
 *  Results:
 *  None.
 *
 *  Side effects:
 *  Memory freed
 *
 *----------------------------------------------------------------------
 */

void TICKET_Shutdown (HTTP_Server *serverPtr)
   {
   HashEntry *entryPtr;
   HashSearch search;
   DString   *dstring;

   DStringFree(&TicketGlobalData(AuthServer));
   DStringFree(&TicketGlobalData(TicketExpHandler));
   DStringFree(&TicketGlobalData(TicketAdrHandler));

   entryPtr = FirstHashEntry(&TicketServerData.SecretsKid, &search);
   while (entryPtr ! = NULL)
    {
    dstring = GetHashValue(entryPtr);
    DStringFree(dstring);
    free(dstring);
    entryPtr = NextHashEntry&search);
    }
  DeleteHashTable(&TicketServerData.SecretsKid);
  DeleteHashtable(&TicketServerData.KeyID);
  DeleteHashTable(&TicketServerData.Domains);
  }
/*
 *
----------------------------------------------------------------------
-
 *
 * TICKET_AddRegion Commands --
 *
 *         Add TICKET region commands for authentication/authorization
decisions.
 *
 * Results:
          None.
```

```
 *
 * Side effects:
          Commands added to the region interpreter.
 *
 *----------------------------------------------------------------------
 */

void TICKET_AddRegion Commands (HTTP_Request *reqPtr, Tcl_Interp *interp)
      {
      Tcl_CreateCommand(interp, "RequireSID", TICKET_RequireSidCmd,
                        (ClientData) reqPtr, NULL);
      Tcl_CreateCommand(interp, "RequireTicket", TICKET_RequireTicketCmd,
                        (ClientData) reqPtr, NULL);
      }


/*
 *----------------------------------------------------------------------
 *
 * TICKET_GetCGIVariables --
 *
 *    Add TICKET CGI variables to the CGI variable table.
 *
 * Results:
          None.
 *
 * Side effects:
          Extends the CGI variable hash table.
 *
 *----------------------------------------------------------------------
 */

void TICKET GetCGIVariables(HTTP_Request *req)
      {
      TICKET_Request *ticketPtr = (TICKET_Request *)
HT_GetReqExtData(req.TicketS

      /*
       * If there's no extension data, then we're not doing a ticket.  Just
return
      */

      if (ticketPtr == NULL)
        return)\;
```

```
      if  (DStringLength(&ticketPtr->rawUrl) ! = 0)
           HT_AddCGIParameter(req, "TICKET_URL", DStringValue(&ticketPtr-
>rawUrl), FA
      if (DStringLength (&ticketPtr->sid) != 0)
           HT_AddCGIParameter(req, "TICKET_SID", DStringValue(&ticketPtr-
>sid), FALSE
      if (DStringLength(&ticketPtr->fields) != 0)
           HT_AddCGIParameter(req, "TICKET_FIELDS", DStringValue(&ticketPtr-
>fields).
      if (DStringLength(&ticketPtr->signature) != 0)
      HT-AddCGIParameter(req, "TICKET_SIGNATURE". DStringValue(&ticketPtr-
>signa
      }/*
 *---------------------------------------------------------------------------
 *
 *TICKET_GetUrl
 *
 *         Return the orignal url (with sid)
 *
 * Results:
 *         The URL.
 *
 * Side effects:
           None.
 *
 *---------------------------------------------------------------------------
 */
char * TICKET_GetUrl(HTTP_Request *reqPtr)
      {
      TICKET_Request *ticketPtr;

      ticketPtr = (TICKET_Request *)
           HT_GetReqExtData(reqPtr, TicketServerData.ticketExtensionId);
      if ((ticketPtr != NULL) &&
            (DStringLength(&ticketPtr->rawUrl) != 0))
           return DStringValue(&ticketPtr->rawUrl);
      else
           return DStringValue(&reqPtr->url);
      }


/*
 *---------------------------------------------------------------------------
 *
 * TICKET_ConfigCheck
 *
 *         Perform late configuration checks
```

```
 *
 * Results:
 *
 *
 * Side effects:
 *         Possible message loged/printed, and program exit'd.
 *
 *----------------------------------------------------------------------
 */
void TICKET_ConfigCheck()
      {
      HashEntry *entryPtr;
      int kid;

      if ((TicketGlobalData(EnableSid) & -0x1) != 0)
           {
           LogMessage(LOG_ERR, "EnableSid must be 0 or 1");
           exit (0);
           }
      if (!(TicketGlobalData(EnableSid))) return;

      kid = TicketGlobalData(CurrentSecret);
      if (kid && kid_mask) != kid)
           {
           LogMessage(LOG-ERR; "CurrentSecret %d is invalid", kid);
           exit(0);
           }

      entryPtr = FindHashEntry(&TicketServerData.SecretsKid, (void *) kid);

      if(entryPtr == NULL)
           {
           LogMessage(LOG_ERR), "No secret defined for CurrentSecret %d", kid;
           exit(0);

      if ((TicketGlobalData(FreeArea) & ~0x255) != 0)
           {
           LogMessage(LOG_ERR, "FreeArea must be between 0 and 255");
           exit(0);
           }

      if ((TicketGlobalData(EnableSidTicket) & -0x1) != 0)
           {
           LogMessage(LOG_ERR, "EnableSidTicket must be 0 or 1");
           exit (0);
           }
```

```
        if ((TicketGlobalData(EnableTicket) & -0x1) !- 0);
             {
             LogMessage(LOG_ERR, "EnableTicket must be 0 or 1");
             exit(0);
             }

        if ((TicketGlobalData(EnableLocalAuth) & -0x1) != 0)
             {
             LogMessage(LOG_ERR, "EnablLocalAuth must be 0 or 1");
             exit (0);
             }
        }
/*
 *----------------------------------------------------------------------
 *
 * TICKET_DebugHooks
 *
 *     Check for debug hooks and execute if found.
 *
 * Results:
 *     None.
 *
 * Side Effects:
 *     None.
 *
*----------------------------------------------------------------------
 */
 tatic void TICKET_DebugHooks(ClientData clientData, char *suffix,
                              HTTP_Request "reqPtr)
      {
      if(strcmp(suffix, "/ticketstatus") == 0)
        {
        DumpStatus(reqPtr);
        HT_FinishRequest(reqPtr);
        return;
       }
    HTTP_Error(reqPtr, NOT_FOUND, "access denied due to poorly formed url");
    HT_FinishRequest(reqPtr);
    return;
    }
/*
 *----------------------------------------------------------------------
 *
 * DumpStatus --
 *
 *     Dump the server's ticket stat's
 *
```

```
 * Results:
 *     None.
 *
 * Side effects:
 *     None.
 *
 *----------------------------------------------------------------------
 */
#define BUFSIZE 1024
static void DumpStatus(HTTP_Request *reqPtr)
{
    HTTP_Server *serverPtr = reqPtr->serverPtr;
    char tmp[BUFSIZE], timeStr[BUFSIZE];
    struct utsname sysinfo;
    time_t uptime;
    int hours;

    HTTP_BeginHeader(reqPtr, "200 OK);
    HTTP_SendHeader(reqPtr, "Content-type: text/html", NULL);
    HTTP_EndHeader(reqPtr);
    HTTP_Send(reqPtr, "<title>WebServer Ticket Status</title>",
                        "<hl>WebServer Ticket Status</hl>:, NULL);

    HTTP_Send(reqPtr, "<p><hr>><p><h2>Ticket Log</h2>", "<p><pre>\n", NULL);

    sprintf(tmp, "    <b>%s:  </b>  %d\n", "Number of access          ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf(tmp, "    <b>%s:  </b>  %d\n", "Number of SID URL's       ", Ticket
    HTTP_Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Valid SID's     ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Expired SID's   ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "   <b>%s:  </b>  %d\n:, "Number of Invalid SID's   ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of XDomain accesses ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of Local Redirects  ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of Remote Redirects ", Ticket
    HTTP)Send(reqPtr, tmp, NULL);
    sprintf (tmp, "  <b>%s:  </b>  %d\n:, "Number of No Auth servers  ", Ticket

    HTTP_Send(reqPtr, tmp, "</pre>", NULL);

    uptime = time (NULL) = serverPtr->started;
    uname(&sysinfo);
```

```
    strftime(timeStr, BUFSIZE, "%A, %d-%b-%y %T",
        localtime(serverPtr->started));

    springf(tmp, "Server runing on <d>%s</b> (%s %s) port %d, has been up \
                    since %s.<p>", sysinfo.nodename, sysinfo.sysname,
                      sysinfo.release, serverPtr->server_port, timeStr);
    HTTP_Send(reqPtr, tmp, NULL);

    sprintf(tmp, "    <b>Number of connections:        </b> %d\n",
              serverPtr->numConnects);
    HTTP_Send(reqPtr, tmp, "<p><pre>\n", tmp, NULL);
    sprintf(tmp, "    <b>Number of HTTP requests:      </b> %d\n",
    HTTP_Send(reqPtr, tmp, "</pre><p>", NULL);

    hours = max(uptime / 3600, 1);
    sprintf(tmp, "This server is averaging <b>%d</b> requests per hour.<p>",
              serverPtr->numRequests/hours);
    HTTP_Send(reqPtr, tmp. NULL);

    DumpRusage(reqPtr);
/*    DumpConnections(reqPtr); */

    DNS_DumpStats(reqPtr);

    HTTP_Send(reqPtr, "<p><hr><address>", DStringValue(&ht_serverSoftware),
              "</address>\n", NULL);

    reqPtr->done = TRUE;

}
#undef BUFSIZE
```

CLAIMS

What is claimed is:

1. A method of processing service requests from a client to a server system through a network comprising:
forwarding a service request from the client to the server system;
returning a session identifier from the server system to the client; and
appending the session identifier to the request and subsequent service requests from the client to the server system within a session of requests.

2. A method as claimed in Claim 1 wherein the server system tracks an access history of sequences of service requests within the session of requests.

3. A method as claimed in Claim 2 wherein the server system tracks the access history to determine service requests leading to a purchase made within the session of requests.

4. A method as claimed in Claim 1 wherein the server system counts requests to particular services exclusive of repeated requests from a common client.

5. A method as claimed in Claim 1 wherein the server system maintains a database relating customer information to access patterns, the information including customer demographics.

6. A method as claimed in Claim 1 wherein the server system subjects the client to an authorization routine prior to issuing the session identifier and the session identifier is protected from forgery.

7. A method as claimed in Claim 6 wherein the server system comprises plural servers including an authentication server which provides session identifiers for service requests to multiple servers.

8. A method as claimed in Claim 7 wherein:

a client directs a service request to a first server which is to provide the requested service;

the first server checks the service request for a session identifier and only services a service request having a valid session identifier, and where the service request has no valid identifier:

the first server redirects the service request from the client to the authorization server;

the authorization server subjects the client to the authorization routine and issues the session identifier to be appended to the service request to the first server;

the client forwards the service request appended with the session identifier to the first server; and

the first server recognizes the session identifier and services the service request to the client; and

the client appends the session identifier to subsequent service requests to the server system and is serviced without further authorization.

9. A method as claimed in Claims 1 or 7 wherein the session identifier includes a user identifier.

10. A method as claimed in Claims 1 or 7 wherein the session identifier includes an expiration time for the session.

11. A method as claimed in Claim 7 wherein the session identifier provides access to a protected domain to which the session has access authorization.

12. A method as claimed in Claim 11 wherein the session identifier is modified for access to a different protected domain.

13. A method as claimed in Claim 7 wherein the session identifier provides a key identifier for key management.

14. A method as claimed in Claims 1 or 7 wherein the server system records information from the session identifier in a transaction log in the server system.

15. A method as claimed in Claims 1 or 7 wherein communications between the client and server system are according to hypertext transfer protocol and the session identifier is appended as part of a path name in a uniform resource locator.

16. A method as claimed in Claim 15 wherein the client modifies the path name of a current uniform resource locator using relative addressing and retains the session identifier portion of the path name unmodified for successive requests in the session.

17. A method as claimed in Claim 1 or 7 further comprising excluding requests made to information from the client within a defined period of time.

18. A method of processing service requests from a client to a server system through a network comprising:
responding to a request for a document received from the client through the network;
appending a session identifier, which includes a user identification, to the request; and
returning the requested document wherein the document is customized for a particular user based on the user identification of the session identifier.

19. A method of processing service request for a document received from a client through network in which the document has been purchased by a user comprising:
responding to a request for a document received from a client through the network in which the document has been purchased by the user;
appending an authorization identifier to the request; and
returning the requested document if the authorization identifier indicates that the user is authorized to access the document.

20. A method as claimed in Claim 19, wherein the authorization identifier is encoded within a session identifier which is appended to the request.

21. A method of processing service requests from a client to a server system through a network comprising:
responding to a request for a document received from a client through the network;
appending a user identifier to the request;

returning the requested document to the client, and;

charging the user identified in the identifier for access to the document.

22. A method as claimed in Claim 21, wherein a user identifier is encoded within a session identifier which is appended to the request.

23. A method of processing service requests from a client to a server system through a network comprising:

forwarding a service request from the client to the server system; and

appending a session identifier to the request and subsequent service requests from the client to the server system within a session of requests.

24. An information system on a network comprising:

means for receiving service requests from clients and for determining whether a service request includes a session identifier;

means for providing the session identifier in response to an initial service request in a session of requests; and

means for servicing service requests from a client which include the session identifier, the subsequent service request being processed in the session.

25. An information system as claimed in Claim 24 wherein the means for providing the session identifier is in a server system which services the requests.

26. An information system as claimed in Claim 23 further comprising an authorization routine for authorizing

the client prior to issuing the session identifier and means for protecting the session identifier from forgery.

27. An information system as claimed in Claim 24 further comprising a transaction log for recording information from the session identifier.

28. An information system as claimed in Claim 24 further comprising means for tracking access history of sequences of service requests within the session.

29. An information system as claimed in Claim 24 further comprising means for counting requests to particular services exclusive of repeated requests from a common client.

30. An information system as claimed in Claim 24 further comprising a database relating customer information to access patterns, the information including customer demographics.

31. An information system as claimed in Claim 25 wherein communications between the client and server system are according to hypertext transfer protocol and the session identifier is appended as part of a path name in a uniform resource locator.

32. An information server on a network comprising:

means for responding to requests for hypertext pages received from a client through the network by returning the requested hypertext pages to the client;

means for responding to further requests derived from links in the hypertext pages; and

means for tracking the further requests derived from a particular hypertext page.

33. An information server as claimed in Claim 32 wherein the requests include a common session identifier and the server tracks requests within a session of requests.

34. An information server as claimed in Claim 32 further comprising a data base relating customer demographics to access patterns.

35. A method of providing access to information pages from a client to a server system through a network comprising:

providing a telephone number at the client;

mapping the telephone number to a target page identifier using a translation database;

requesting information described by the page identifier from the server system; and

displaying a page identified by the page identifier at the client.

36. A method of providing access to information pages from a client to a server system through a network comprising:

providing a descriptor at the client;

mapping the descriptor to a target page identifier using a translation database;

requesting at the client information described by the page identifier from the server system without further user action; and

displaying a page identified by the page identifier at the client.

37. A method as claimed in Claims 35 or 36 wherein the translation database resides in the server system which returns a uniform resource loctor in a REDIRECT command to the client to cause the client to request the information using the uniform resource locator.

38. A method as claimed in Claim 36 wherein the descriptor comprises a telephone number.

39. A method as claimed in Claim 36 wherein the descriptor comprises a descriptive term.

40. A method as claimed in Claim 39 wherein the term includes a company name.

41. A method as claimed in Claim 39 wherein the term includes a product name.

42. A method as claimed in Claim 39 wherein the term is identified by phonetic mapping.

43. A method as claimed in Claims 35 or 38 wherein the target page identifier describes a controlled page.

44. A method as claimed in Claims 35 or 36 wherein the target page identifier is a uniform resource locator.

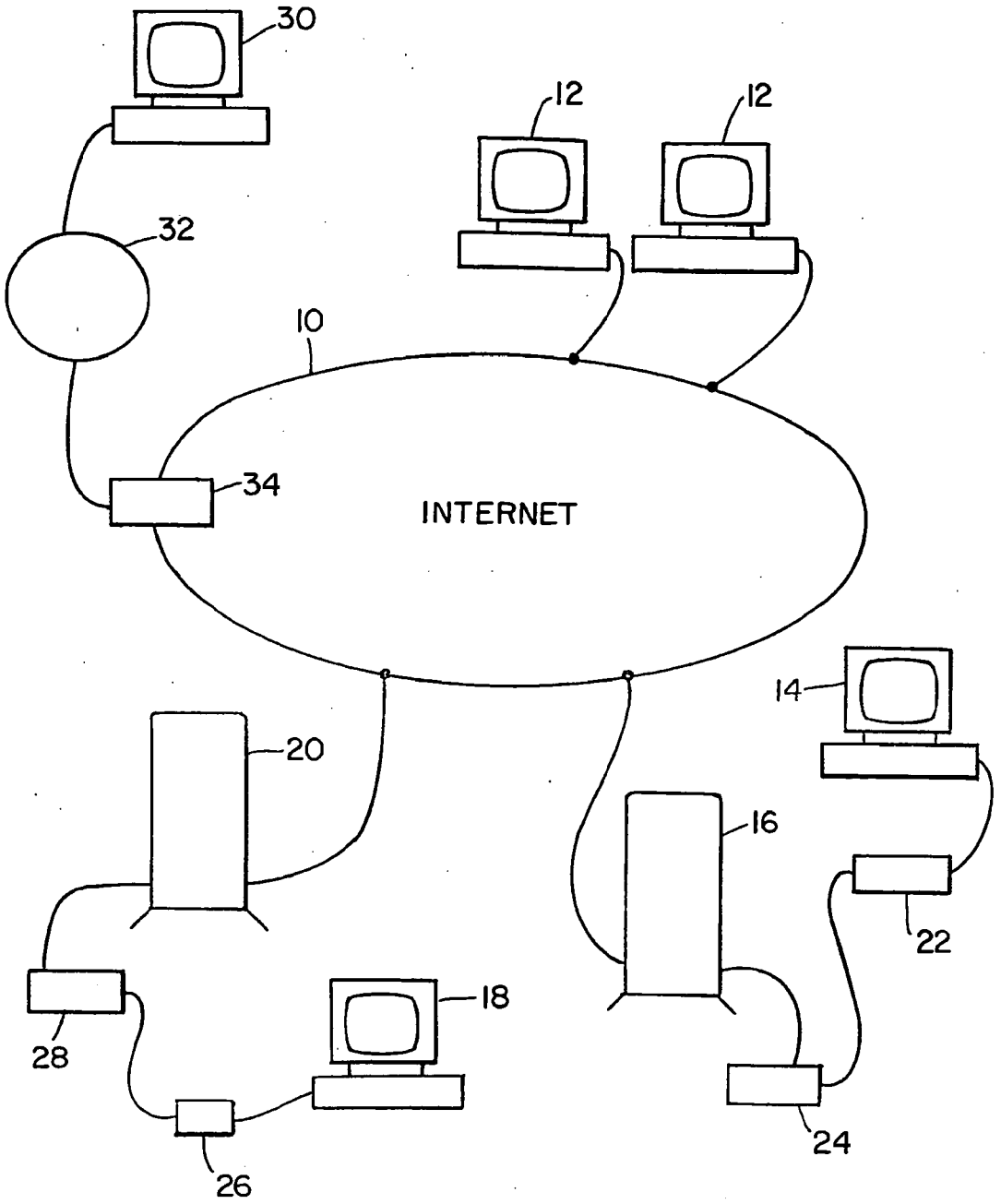


FIG. 1

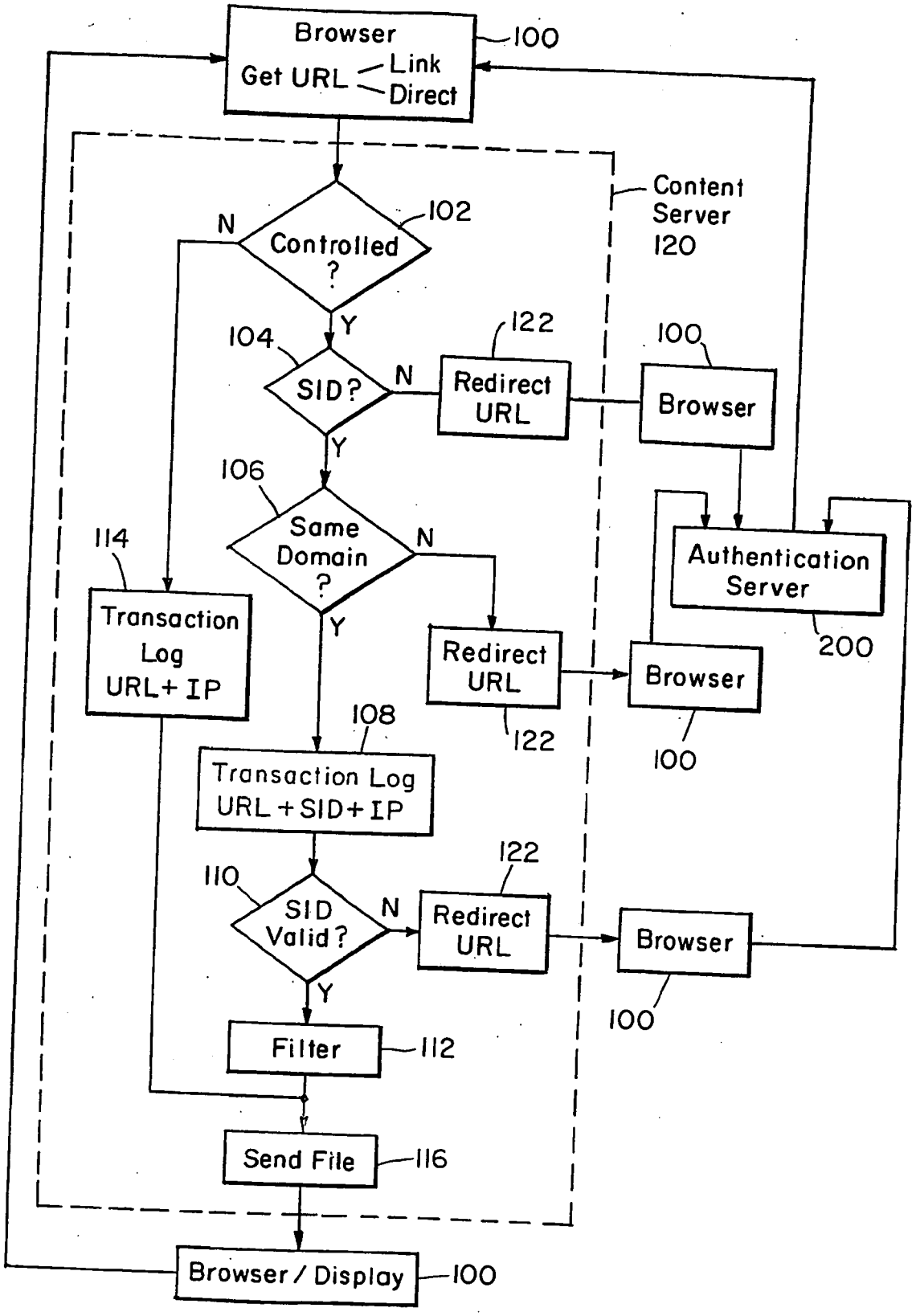

Browser
Get URL
Link
Direct
100
Content Server 120
Controlled ?
102
N
Y
104
SID?
N
Y
122
Redirect URL
100
Browser
106
Same Domain ?
N
Y
114
Transaction Log URL+IP
Redirect URL
122
Browser
100
Authentication Server
200
108
Transaction Log URL+SID+IP
110
SID Valid?
N
Y
122
Redirect URL
Browser
100
Filter
112
Send File
116
Browser / Display
100

FIG. 2A

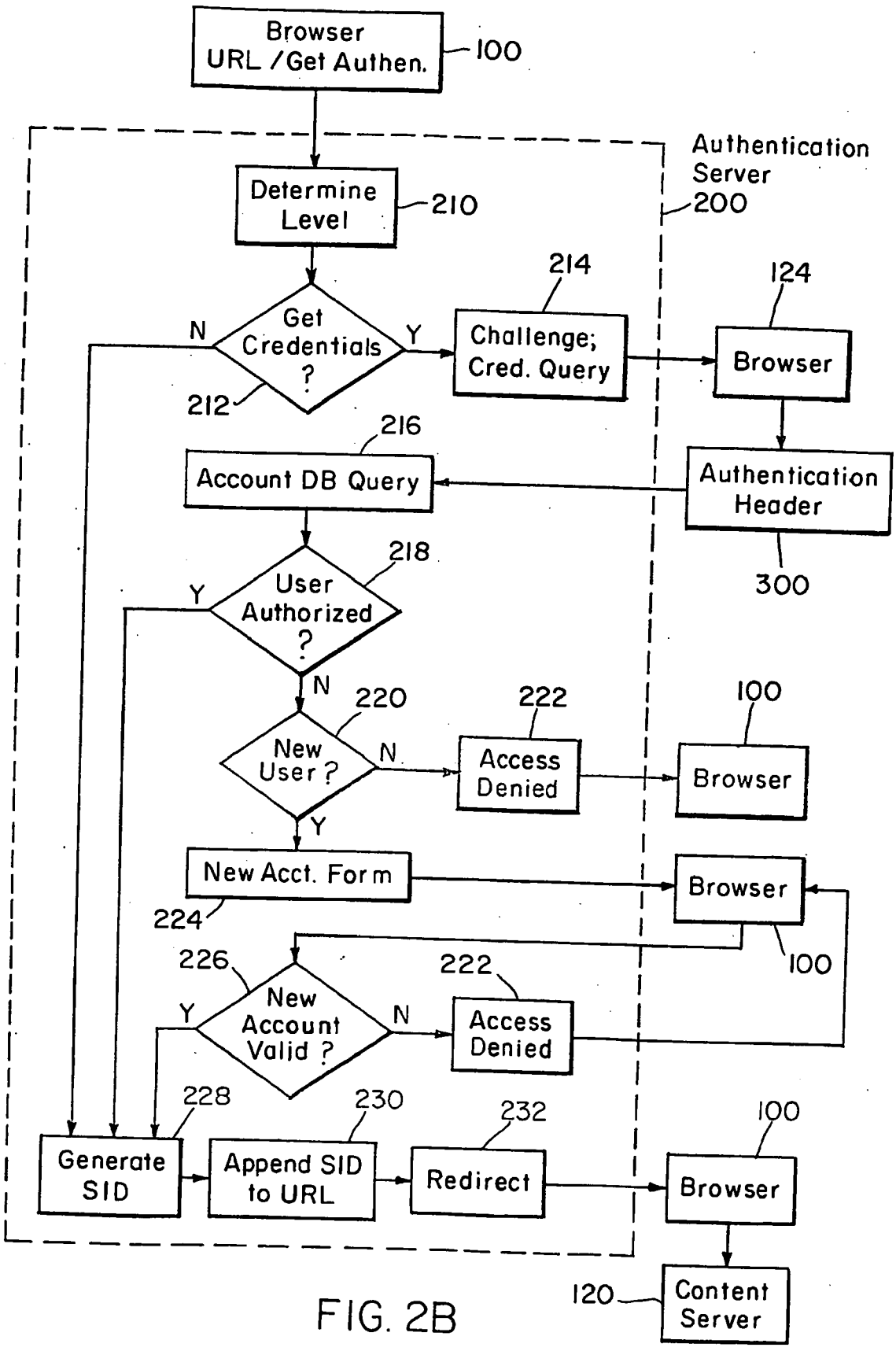

Browser URL /Get Authen.
100
Authentication Server
200
Determine Level
210
214
124
Get Credentials ?
N
Y
Challenge; Cred. Query
Browser
212
216
Account DB Query
Authentication Header
300
218
User Authorized ?
Y
N
220
222
100
New User ?
N
Access Denied
Browser
Y
New Acct. Form
Browser
224
100
226
222
New Account Valid ?
Y
N
Access Denied
228
230
232
100
Generate SID
Append SID to URL
Redirect
Browser
120
Content Server

FIG. 2B

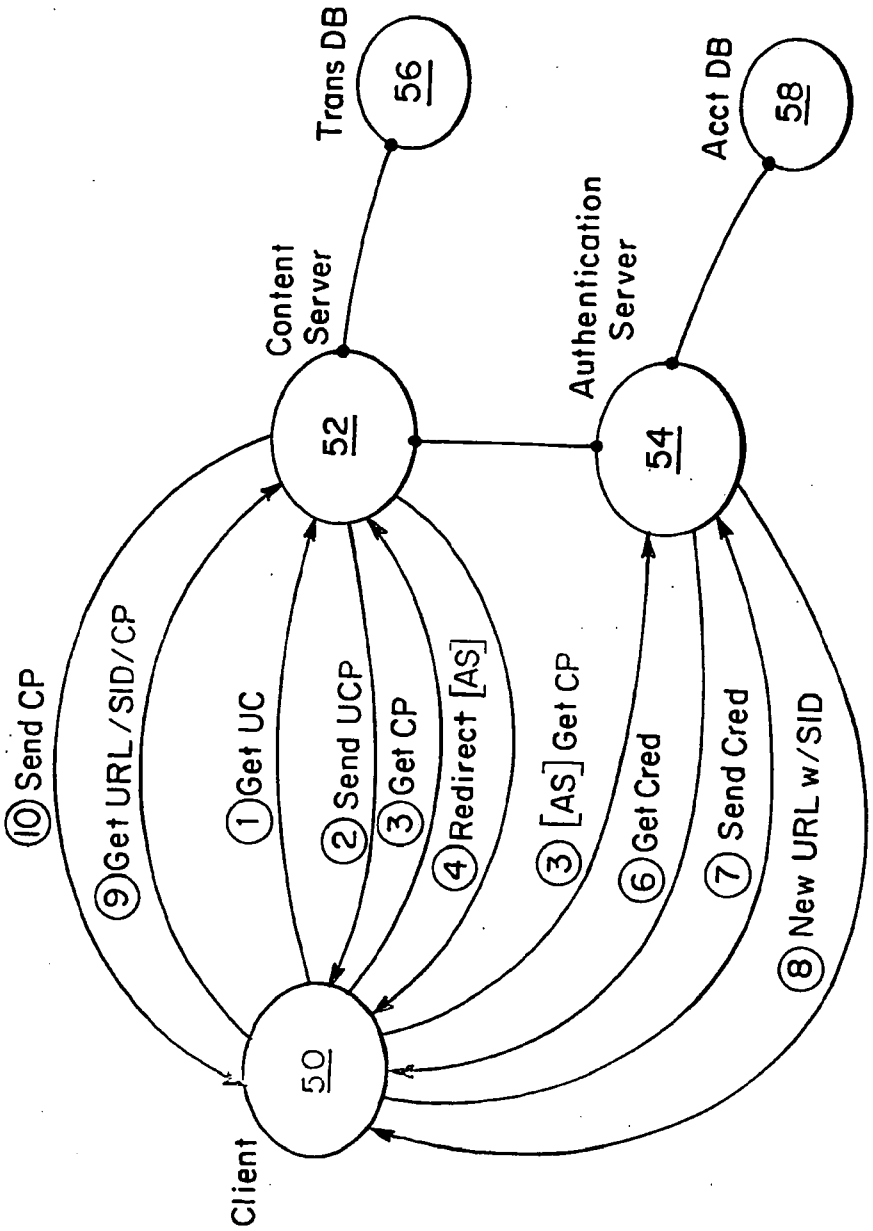

Trans DB
56
Acct DB
58
Content Server
52
Authentication Server
54
Client
50
① Get UC
② Send UCP
③ Get CP
④ Redirect [AS]
③ [AS] Get CP
⑥ Get Cred
⑦ Send Cred
⑧ New URL w/SID
⑨ Get URL/SID/CP
⑩ Send CP

FIG. 3

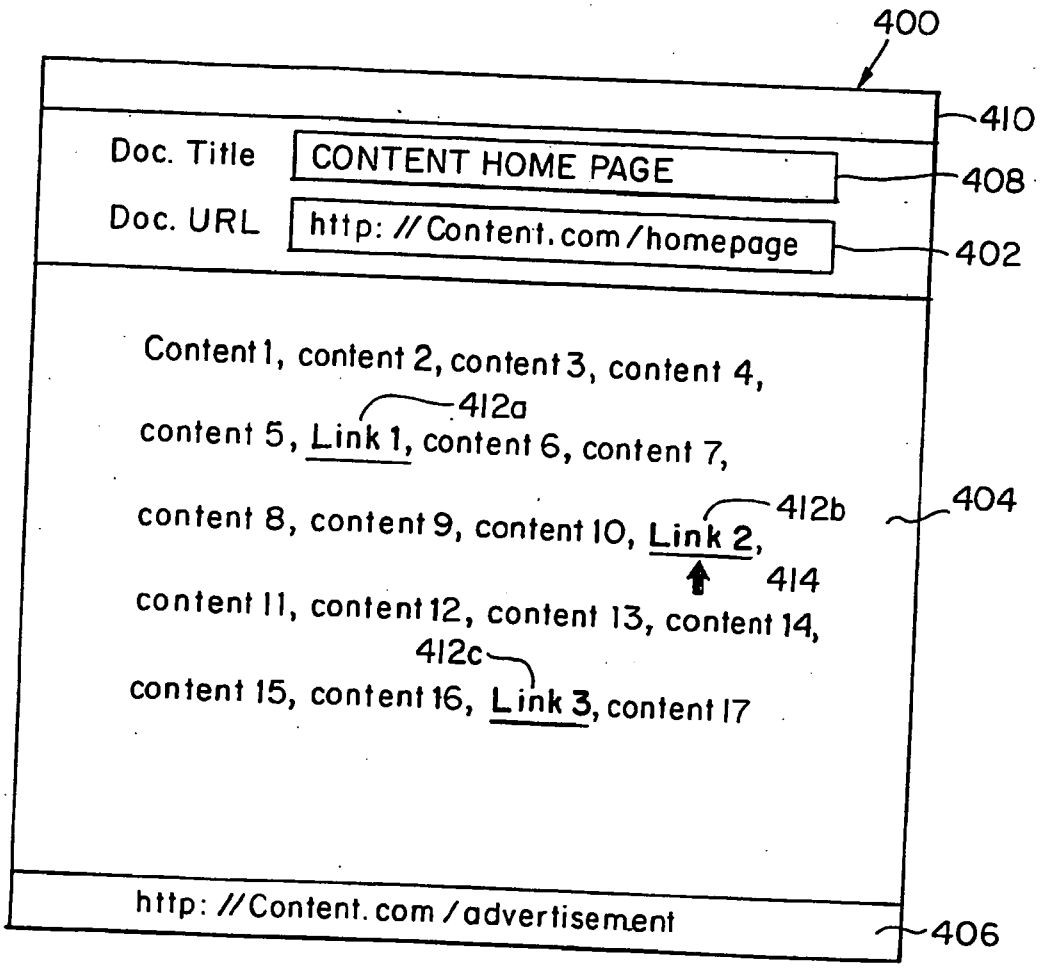

400
410
Doc. Title
CONTENT HOME PAGE
408
Doc. URL
http://Content.com/homepage
402
Content 1, content 2, content 3, content 4,
412a
content 5, Link 1, content 6, content 7,
content 8, content 9, content 10, Link 2,
412b
404
414
content 11, content 12, content 13, content 14,
412c
content 15, content 16, Link 3, content 17
http://Content.com/advertisement
406

FIG. 4

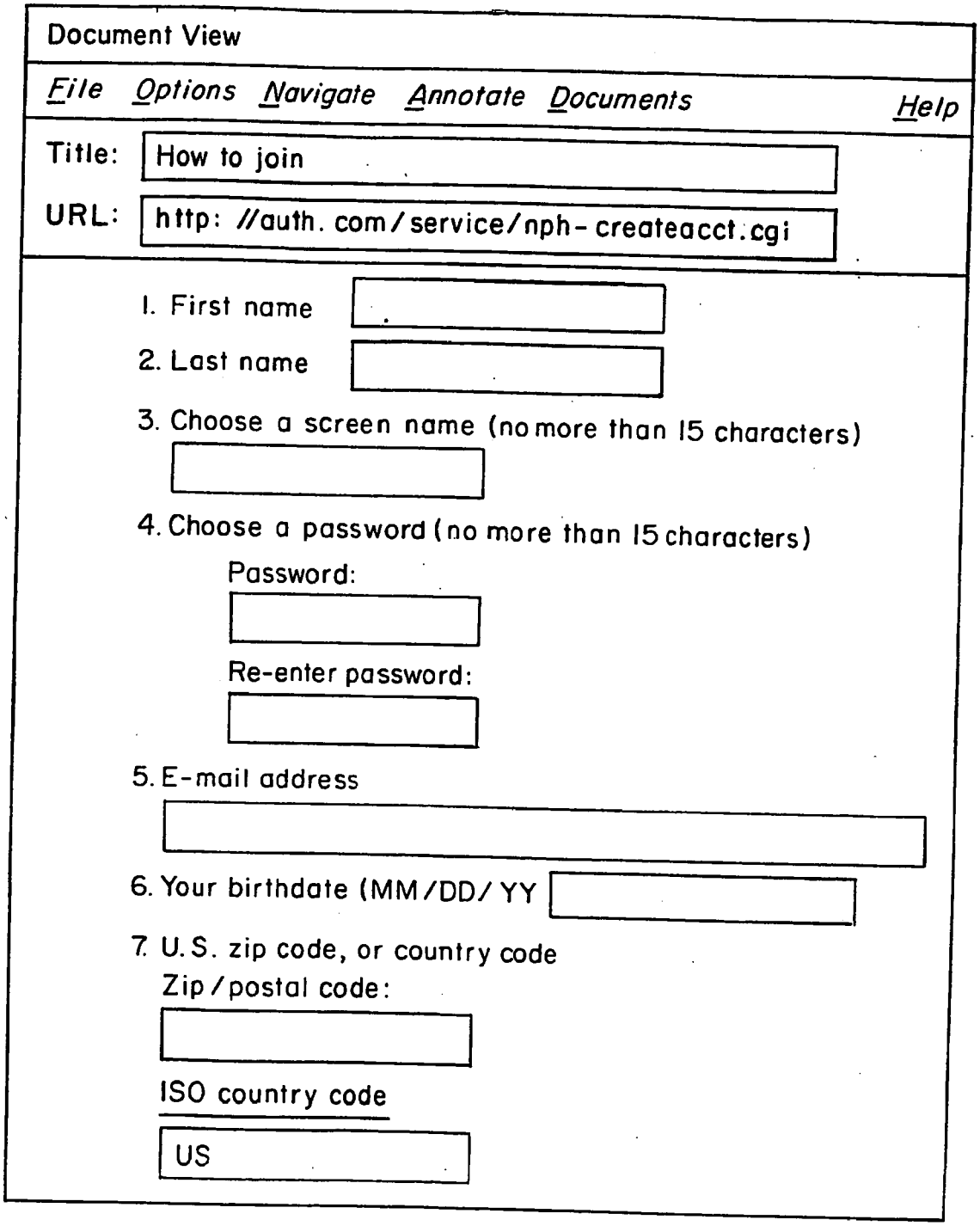

Document View
File Options Navigate Annotate Documents Help
Title: How to join
URL: http: //auth. com/service/nph-createacct.cgi
1. First name
2. Last name
3. Choose a screen name (no more than 15 characters)
4. Choose a password (no more than 15 characters)
Password:
Re-enter password:
5. E-mail address
6. Your birthdate (MM/DD/YY
7. U.S. zip code, or country code
Zip/postal code:
ISO country code
US

FIG. 5

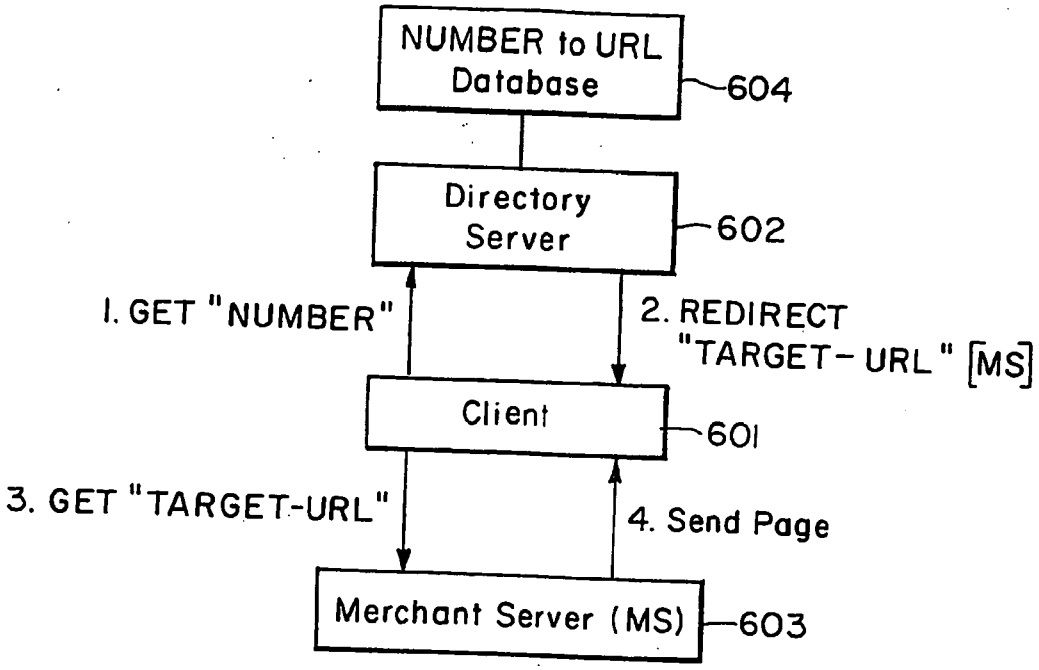

NUMBER to URL Database
604
Directory Server
602
1. GET "NUMBER"
2. REDIRECT "TARGET-URL" [MS]
Client
601
3. GET "TARGET-URL"
4. Send Page
Merchant Server (MS)
603

FIG. 6

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 13/14, 13/42, H04L 12/46, 29/02 — A1

(11) International Publication Number: **WO 97/11429**

(43) International Publication Date: 27 March 1997 (27.03.97)

(21) International Application Number: PCT/US96/14988

(22) International Filing Date: 13 September 1996 (13.09.96)

(30) Priority Data:
08/531,371 20 September 1995 (20.09.95) US

(71) Applicant: INFONAUTICS CORPORATION [US/US]; Suite 900, 900 West Valley Road, Wayne, PA 19087 (US).

(72) Inventors: GRABER, Terry, E.; 108 Brookhollow Drive, Downingtown, PA 19335 (US). KOPELMAN, Joshua; Unit 906, 611 W. King Street, Charleston Greene, Malvern, PA 19355 (US). WATKEYS, Edwin, Howel, III; 308 South Pennsylvania Avenue, North Wales, PA 19454 (US). WEINBERGER, Marvin, I.; 630 Pennfield Avenue, Havertown, PA 19083 (US).

(74) Agent: GOLUB, Daniel, H.; Reed Smith Shaw & McClay, 2500 One Liberty Place, 1650 Market Street, Philadelphia, PA 19103-7301 (US).

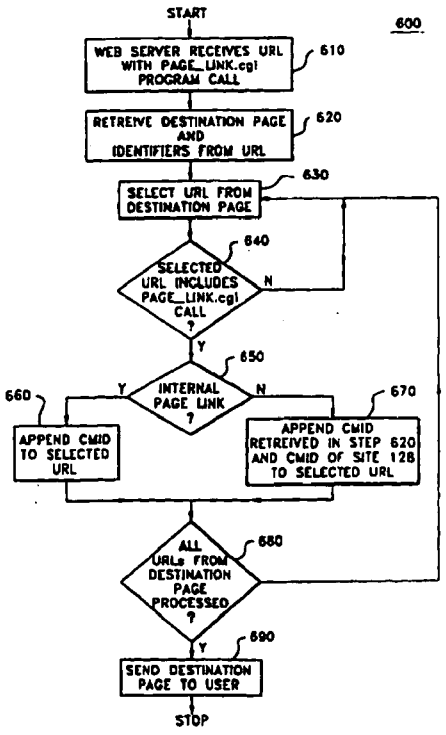
(81) Designated States: AU, CA, CN, JP, MX, NZ, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: REDIRECTING A USER TO A NEW WORLD WIDE WEB LOCATION USING RELATIVE UNIVERSAL RESOURCE LOCATORS

(57) Abstract

A method and apparatus for redirecting a user from a first location on the WWW to a second location on the WWW, wherein relative URL addressing is used during the redirecting process. A signal is received from the first location indicating that the user wishes to move from the first location on the WWW to the second location on said WWW (610, 620). In response to the signal, a current URL representing an address of the first location on the WWW and a destination URL portion representative of an address of the second location on the WWW are passed to a redirecting means (630, 640). The current URL includes first and second portions. A destination URL is formed with redirecting means by substituting the destination URL portion in place of the second portion in the current URL, wherein the destination URL represents a relative address of the second location on the WWW (660, 670). The user is then moved from the first location on the WWW to the second location on the WWW in accordance with the destination URL formed by the redirecting means (690).



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgystan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SG	Singapore
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LR	Liberia	SZ	Swaziland
CS	Czechoslovakia	LT	Lithuania	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	LV	Latvia	TJ	Tajikistan
DK	Denmark	MC	Monaco	TT	Trinidad and Tobago
EE	Estonia	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	UG	Uganda
FI	Finland	ML	Mali	US	United States of America
FR	France	MN	Mongolia	UZ	Uzbekistan
GA	Gabon	MR	Mauritania	VN	Viet Nam

REDIRECTING A USER TO A NEW WORLD WIDE WEB LOCATION USING RELATIVE UNIVERSAL RESOURCE LOCATORS

Field of the Invention

The present invention is directed to on-line computer systems for delivering information and computer services to users coupled to such systems. More particularly, the present invention is directed to an automated system for capturing information representing the identity of an entity that has directed a user to an on-line system. Still more particularly, the present invention is directed to a system for tracking user paths on the world wide web (WNW).

Background of the Invention

On-line computer services such as, for example, on-line information retrieval services, on-line travel reservation services, or on-line stock trading services, receive new subscribers from various sources. New subscribers are typically directed to an on-line service by advertisements placed by the on-line service itself, through word-of-mouth referrals given by existing system subscribers, or by third party computer system marketers (referred to hereinafter as co-marketers) of the on-line service. Different co-marketers may typically use different media for promoting a particular on-line computer service. For example, a magazine acting as a co-marketer for an on-line service might use a magazine advertisement, which includes a free software disk for accessing the on-line service, to promote the on-line service. Alternatively, various directory services available on the WWW such as, for example, the Yaho® or Web Crawler® directory services, might use a listing on their directory pages and a link to a WWW page associated with an on-line service, to direct potential new subscribers to an on-line service. Thus, new subscribers can be directed to the same on-line service from different co-marketers and through different marketing channels. It would be desirable to be able to capture and track the co-marketing source which directed each new subscriber to an on-line service. In addition, it would be desirable to capture and track the co-marketing source which directed a new subscriber to an on-line service in a manner which required no participation or intervention from the new subscriber.

When a user navigates through various sites during a session on the WWW, the navigational history reflecting the past locations traversed by the user during the session is typically lost as the user moves from one site to the next site. Thus, unless the user were

to manually track the various sites traversed during a world wide web session, it would be difficult for the user, or for any service monitoring the user, to know the identity of any previous world wide web site traversed by the user during a session. It would be desirable to have a system for attaching navigational history information to a user traversing the WWW so that a current web site could determine electronically at least the previous WWW site visited by the user.

Universal resource locators are often used to direct users through various pages at a site on the world wide web. There are two different techniques for specifying addresses using universal resource locators. In a first technique, known as fully specified addressing, the full string associated with a universal resource locator is specified each time a user moves from one web page to the next web page. In the second technique, known as relative addressing, only information representing the root directory or the current directory (or subdirectory) of the user is specified as a user moves from one web page to the next web page. One drawback of using relative universal resource locator addressing is that it is impossible to move "up a directory tree" using such addressing, without specifying the root directory. UNIX symbolic links may be used in specifying a particular root directory. However, when relative addressing is used, it is impossible to carry this UNIX symbolic link information forward as a user moves from page to page. It would be desirable if this limitation of relative universal resource locator addressing could be ameliorated, such that the UNIX symbolic link information could be retained during the relative addressing of web pages.

It is therefore an object of the present invention to provide a unified system for capturing and tracking a co-marketing source which directed a new subscriber to an on-line service.

It is a further object of the present invention to provide a system for capturing and tracking information identifying a co-marketing source which directed a new subscriber to an on-line service, which requires no participation or intervention from the new subscriber.

It is a still further object of the present invention to provide a system for attaching navigational history information to a user traversing the world wide web so that a current web site could determine electronically at least the previous world wide web site visited by the user.

It is a still further object of the present invention to provide a system which could be used in conjunction with relative universal resource locator addressing, which permitted a user in a particular directory at a web site to move up a directory tree.

These and other objects of the invention will become apparent from the description of the invention which follows.

Summary of the Invention

The present invention is directed to an apparatus for capturing and storing a co-marketer identification symbol representing an identity of an entity that has referred a user on a user station to a computer service, wherein the user station is coupled to the computer service by a communications path. A database is provided for storing a plurality of user records. Each of the user records includes a user identification field for storing information uniquely associating each of the user records with a user, and a co-marketer identification field for storing identity information representing the identity of an entity that directed the user to the computer service. An enrollment means is coupled to the communications path and the database, and is provided for enrolling a user on the computer service. The enrollment means includes means for determining a co-marketer that directed the user to the computer service, and means for assigning a unique user identification number to the user. The enrollment means further includes means for storing a co-marketer identification symbol representative of a co-marketer and the unique user identification number of a user in the co-marketer identification and user identification fields, respectively, of one of the user records.

In accordance with a further aspect, the present invention is directed to a method and apparatus for tracking the navigation path of a user that has been directed to a second site on the WWW from a first site on the WWW. The first site has a universal resource locator (URL) symbol for uniquely identifying an address of the first site on the WWW, and the second site has a URL symbol for uniquely identifying an address of the second site on the WWW. A composite URL symbol is received at the second WWW site when the user is directed from the first site to the second site. The composite URL symbol has a first portion corresponding to the URL symbol of the second site, and a second portion that includes information corresponding to the identity of the first site. The information representative of the identity of the first site is captured at the second WWW site from the second portion of the composite URL. The identity of the first WWW site is then determined at the second WWW site by comparing information from the second portion of

the composite URL to a table having a plurality of entries each of which is representative of a known WWW site.

In accordance with a still further aspect, the present invention is directed to a method and apparatus for redirecting a user from a first location on the WWW to a second location on the WWW, wherein relative URL addressing is used during the redirecting process. A signal is received from the first location indicating that the user wishes to move from the first location on the WWW to the second location on said WWW. In response to the signal, a current URL representing an address of the first location on the WWW and a destination URL portion representative of an address of the second location on the WWW are passed to a redirecting means. The current URL includes first and second portions. A destination URL is formed with redirecting means by substituting the destination URL portion in place of the second portion in the current URL, wherein the destination URL represents a relative address of the second location on the WWW. The user is then moved from the first location on the WWW to the second location on the WWW in accordance with the destination URL formed by the redirecting means.

In accordance with a still further aspect, the present invention is directed to a method and apparatus for tracking the navigation path of a user that has been directed to a second site on the WWW from a first site on the WWW. A URL is received at the second WWW site when the user is directed from the first site to the second site. At the second WWW site, information representative of an identity of the first WWW site is captured by identifying a first code in the URL. A destination web page is determined for the user, and a revised destination web page is formed by inserting a second code representative of the identity of the first WWW site into at least one selected web page link associated with the destination web page. The revised destination web page is then transmitted to the user.

Brief Description of the Drawings

In order that the manner in which the above-recited and other advantages and objects of the invention are obtained and can be appreciated, a more particular description of the invention briefly described above will be rendered by reference to a specific embodiment thereof which is illustrated in the appended drawings. Understanding that these drawings depict only a typical embodiment of the invention and are not therefore to be considered limiting of its scope, the invention and the presently understood best mode thereof will be

described and explained with additional specificity and detail through the use of the accompanying drawings.

Figure 1 is a block diagram showing a system for enrolling new users on an on-line system and capturing co-marketing information associated with such new users, in accordance with a preferred embodiment of the present invention.

Figure 2 is a flow diagram illustrating the operation of a system for enrolling new users on an on-line system and capturing co-marketing information associated with such new users, in accordance with a preferred embodiment of the present invention.

Figure 3 is a diagram of a look-up table for associating UNIX symbolic link information with co-marketers, in accordance with a preferred embodiment of the present invention.

Figure 4 is a diagram illustrating preferred data structures for storing a Subscriber Information Directory Table, a Customer Information Directory Table, and first and second Co-Marketer Information Directory Tables, in accordance with a preferred embodiment of the present invention.

Figure 5 is a schematic diagram illustrating the use of UNIX symbolic links and relative URL addressing for moving between locations on the WWW, in accordance with a preferred embodiment of the present invention.

Figure 6 is a flow diagram illustrating the operation of a system for attaching a code representing the navigational history of a user on the WWW onto selected URL page links on a destination web page of a user, in accordance with an alternative preferred embodiment of the present invention.

Figure 7 is a flow diagram illustrating the operation of a system for generating recurring bounty payment records, in accordance with a preferred embodiment of the present invention.

Detailed Description of the Invention

Referring now to Fig. 1, there is shown a block diagram of a system 100 for enrolling new users on an on-line system and capturing co-marketing information associated with such new users, in accordance with a preferred embodiment of the present invention. System 100 includes a first type of user station 102. The user station 102 includes a personal computer (PC) 104 and user software 106 which resides on PC 104. User software

106 includes a graphical user interface (not shown) for facilitating communications between user station 102 and On-Line Service (OLS) 140. OLS 140 represents a computer service such as, for example, an information retrieval service, a travel reservation service, or a stock trading service, which is available on-line to a user of user station 102. User station 102 is coupled to a Fiber Distributed Data Interface (FDDI) 141 in OLS 140 by a communications channel 108. In alternate embodiments, a standard communications bus or a local area network may be substituted for FDDI 141. Communications channel 108 may consist of a communications link formed over a public network such as the Internet. Alternatively, communications channel 108 may consist of a communications link formed between PC 104 and FDDI 141 over a commercial network. Thus, commercial networks such as, for example, the Prodigy® network, the CompuServe® network, or the Microsoft® network, may be used to establish a communications channel 108 for linking PC 104 and FDDI 141. Although in the preferred embodiment of the present invention, element 140 is shown as being an on-line computer service, it will be understood by those skilled in the art that element 140 may alternatively represent any computer service, regardless of whether the service is available on-line.

As explained more fully below, user software 106 is preferably provided to a user of user station 102 by an on-line service co-marketer (CM) and loaded onto PC 104 prior to the time the user of user station 102 attempts to enroll on OLS 140. User software is preferably provided to the user of user station 102 from the CM via a floppy disk, CD-ROM disk, magnetic tape or through a File Transfer Protocol (FTP) site on the Internet. User software 106 preferably includes an embedded co-marketer symbol or code which can be recognized by OLS 140 whenever the user of user station 102 connects to OLS 140. The co-marketer symbol embedded in the user software uniquely represents the identity of the co-marketer that provided user software 106 to the user of user station 102. An example of a co-marketer that might provide user software 106 to a user of user station 102 might include, for example, a magazine publisher that advertises OLS 140 in its magazine and includes a floppy disk with user software 106 together with the magazine advertisement.

System 100 also includes a second type of user station 102a. The user station 102a includes a PC 104a and user software 106a which resides on PC 104a. Like user software 106, user software 106a includes a graphical user interface (not shown) for facilitating communications between user station 102a and On-Line Service (OLS) 140.

However, unlike user station 102, user station 102a is coupled to OLS 140 through the WWW 120. More particularly, user station 102a is coupled to an OLS web server 142 at OLS 140 through the WWW site 128 associated with OLS 140 on the WWW 120.

The navigation history of the user of user station 102a on WWW 120 is shown generally by dotted lines 124, 125 and solid line 126. As shown by dotted line 124, user station 102a was initially coupled to site 122a of a first co-marketer on WWW 120. In the preferred embodiment of the present invention, a page at site 122a includes an advertisement (not shown) for OLS 140. In addition, the advertisement at co-marketer site 122a is preferably such that a user of user station 102a may chose to connect to OLS site 128 simply by "clicking" on the advertisement at WWW site 122a. As explained more fully below in connection with Figure 5, when the user of user station 102a clicks on the advertisement for OLS 140 at WWW site 122a, WWW site 122a forms a special destination URL having two parts. The first part of the destination URL is formed of the URL associated with OLS site 128 (e.g., WWW.OLS.COMM). The second part of the destination URL is formed of a destination filename (e.g., INDEX. HTML) and a UNIX symbolic link (e.g., \CM1) that is prepended to the beginning of the destination filename by the co-marketer (co-marketer #1) associated with WWW site 122a. The symbol or code used to form the UNIX symbolic link (e.g., \CM1) inserted by co-marketer #1 at site 122a is uniquely associated with co-marketer #1 in system 100. The complete destination URL is used to route the user (along dotted line 125) from WWW site 122a of co-marketer #1 to OLS WWW site 128. Upon reaching OLS site 128, the user station 102a is coupled to OLS WWW site 128 by solid line 126, and the complete destination URL formed at site 122a (including the UNIX symbolic link portion of such destination URL) is passed to OLS 140 through OLS web server 142.

In addition to the co-marketer represented by site 122a (co-marketer #1), users may be directed to OLS site 128 on WWW 120 through advertisements (not shown) on pages at the sites of other co-marketers represented on the WWW 120 such as, for example, through an advertisement at WWW site 122b (representing co-marketer #2), or an advertisement at WWW site 122c (representing co-marketer #3). Like the situation described above wherein a user of user station 102a clicks on the advertisement for OLS 140 at WWW site 122a, when the user of user station 102a clicks on the advertisement for OLS 140 at WWW site 122b or 122c, WWW site 122b forms a special destination URL having two

parts. The first part of the destination URL is again formed of the URL associated with OLS site 128 (e.g., WWW.OLS.COMM), and the second part of the destination URL is again formed of a UNIX symbolic link that is prepended to a destination filename. However, if the user has clicked on an advertisement for OLS 140 at site 122b, the UNIX symbolic link (e.g., \CM2) inserted by the co-marketer (co-marketer #2) will be uniquely associated in system 100 with co-marketer #2 and site 122b. Similarly, if the user has clicked on an advertisement for OLS 140 at site 122c, the UNIX symbolic link (e.g., \CM3) inserted by the co-marketer (co-marketer #3) will be uniquely associated in system 100 with co-marketer #3 and site 122c. A complete destination URL formed at either site 122b or 122c may be used as described above in connection with site 122a to route the user from site 122b or 122c to OLS WWW site 128. Although only three co-marketers are shown in Figure 1 for directing users from the WWW sites of such co-marketers to OLS site 128, it will be understood by those skilled in the art that more than three co-marketing sites may be used in conjunction with the present invention for directing users to OLS site 128 on WWW 120.

In the preferred embodiment of system 100, OLS 140 will accept a user that has been routed to OLS site 128 by a co-marketer only if the co-marketer that has done the routing is an authorized co-marketer for OLS 140. In system 100, a co-marketer will be authorized to route users to site 128 only after the co-marketer has been assigned and has received a unique UNIX symbolic link associated with the co-marketer from OLS 140. First and second Co-Marketer Identification Tables are stored respectively on enrollment database 146 and accounting database 144 at OLS 140. As described more fully below in conjunction with Figure 4, each Co-Marketer Identification Table includes a separate record for storing a co-marketer identification code associated with each co-marketer (e.g., co-marketer #1, co-marketer #2, co-marketer #3) that has been authorized by OLS 140 to route users to OLS site 128.

System 100 includes an enrollment server 145 for enrolling new users on OLS 140, and a billing server 143 for generating bounty payment records for issuing bounty payments to authorized co-marketers that have referred users of user stations 102, 102a to OLS 140. For purposes of the present application, the term "server", when used in conjunction with "enrollment" or "billing", is used to refer to a physical machine formed from at least one computer processor having associated memory and software installed thereon for executing the functions to be performed by the server. In the preferred

embodiment of the present invention, the hardware platform used for implementing enrollment server 145 consists of a Tandem Model 4412 computer having 2 processors, 200 MB of memory, a 1 GB system disk, and a 4 GB RAID disk; a flow diagram illustrating the operation of a preferred software system 200 for implementing enrollment server 145 on this hardware platform is shown in Figure 2 and discussed below. In the preferred embodiment of the present invention, the hardware platform used for implementing billing server 143 consists of a Tandem Model 4412 computer having 2 processors, 200 MB of memory, a 1 GB system disk, and a 4 GB RAID disk; a flow diagram illustrating the operation of a preferred software system 700 for implementing billing server 143 on this hardware platform is shown in Figure 7 and discussed below. Although specific hardware is disclosed herein for implementing enrollment server 145 and billing server 143, it will be understood by those skilled in the art that other suitable hardware platforms may alternatively be used to implement servers 143, 145. However, the two hardware systems described above for implementing servers 143, 145 are preferred because these systems allow servers 143, 145 to be hardware-scalable. This "hardware scalability" allows OLS 140 to handle an increasing number of user stations 102, 102a simply by adding further processors to the existing hardware used for servers 143, 145, without modification of the software running on such hardware. In the embodiment shown in Figure 1, enrollment server 145 and billing server 143 are implemented in software on separate machines which are physically distinct from the processor(s) used for implementing OLS session server 147. In alternative embodiments (not shown), enrollment server 145 and billing server 143 may be implemented in software together on a single server or as part of OLS session server 147.

Referring now to Figure 2, there is shown a flow diagram illustrating the operation of a system 200 for enrolling new users associated with user stations 102, 102a onto OLS 140, and capturing co-marketing information associated with such new users, in accordance with a preferred embodiment of the present invention. In step 205, a user station 102 or 102a connects to OLS 140. In the case of a user station 102, the connection to OLS 140 is made by the user station via communications channel 108 directly to FDDI 141; in the case of a user station 102a, the connection is made via OLS site 128 to OLS web server 142. Next, in step 210, the enrollment means 145 determines whether the user which just connected to OLS 140 is a new subscriber to OLS 140. In a preferred embodiment, step 210 is performed either (i) by waiting for the user to issue an enrollment request (from page 514a

described below) to OLS 140, or (ii) by prompting the user to enter a login name and comparing the login name entered by the user to a list of valid login names maintained in enrollment database 146. If the user is a new subscriber to OLS 140, processing proceeds to step 220, where the enrollment means 145 determines how the user connected to OLS 140. More particularly, if the user connected to OLS 140 through web server 142, the enrollment means 145 determines that the user is operating on a user station 102a which is connected to OLS 140 via WWW 120; otherwise, enrollment means 145 determines that the user is operating on a user station 102 which is connected to OLS 140 via communications channel 108.

If a determination is made in step 220 that the user is operating on a user station 102a that connected to OLS 140 through WWW 120, then processing proceeds to step 230 where enrollment means 145 determines a co-marketer identification symbol or code (CM ID) associated with the user station 102a. In this step, the complete destination URL which was passed to OLS web server 142 when the user was directed from a co-marketer site 122a, 122b, 122c to OLS site 128 is retrieved by OLS web server 142, and the second portion of the destination URL, which contains both a UNIX symbolic link and a destination filename (which may be specified implicitly), is then extracted from the complete destination URL. As mentioned above, the UNIX symbolic link embedded in the destination URL uniquely identifies a co-marketer which directed the user from its WWW site to OLS site 128. Next, in step 240, enrollment means 145 attempts to enroll the user in OLS 140. In this step, the enrollment means 145 obtains a co-marketer identification code (CMID) associated with the destination URL using look-up table 300 (shown in Figure 3). For each valid co-marketer in system 100, table 300 has one or more entries representing the second portion of a potential destination URL that might be generated by such a co-marketer. Thus, each entry in table 300 has a record 310 representing a UNIX symbolic link (310a) and destination filename (310b) that may be provided by a valid co-marketer, and a corresponding record 320 representing a CMID associated with the co-marketer assigned to UNIX symbolic link 310a in system 100. If the second portion of the destination URL is not recognized as corresponding to a valid CMID, the enrollment session is terminated. A list of valid (or authorized) CMID's is preferably stored in a Co-Marketer Information Directory Table on enrollment database 146 shown in Figure 4. The Co-Marketer Information Directory Table on enrollment database 146 is formed of a plurality of individual

records 440, each of which contains a field 442 for storing the CMID of a system authorized co-marketer.

Referring still to step 240, if the enrollment means 145 determines that the user has been directed to OLS 140 from an authorized co-marketer, the enrollment means attempts to enroll the user in OLS 140 by assigning the user a unique user identification number and then asking the user to enter various personal information which is then stored in a Subscriber Information Directory Table on enrollment database 146. As shown in Figure 4, the Subscriber Information Directory Table on enrollment database 146 is formed of a plurality of individual records 400, each of which contains several fields for storing information about a particular user. More particularly, for each user enrolled on OLS 140 there is a record 400 with a field 402 for storing the unique user identification number assigned to the user, a field 404 for storing the CM ID of the co-marketer that directed the user to OLS 140, fields 406, 408, 410 for respectively storing the name, address and telephone number of the user, fields 412, 414 for respectively storing the grade level and gender of the user, fields 416, 418 for storing information representing the occupations of the user's parents, field 420 for storing the user's number of siblings, and fields 422, 424, 426, 428 and 430 for respectively storing information representing the type of computer used by the user, the user's modem speed, the display capabilities of the user's display, the size of the memory of the user's PC, and the identity of the communications link (e.g., the Internet, the Prodigy® network, the CompuServe® network, or the Microsoft® network) used for accessing OLS 140.

In addition to storing information about the user in the Subscriber Information Directory Table on enrollment database 146, information about the user being enrolled in step 240 is also stored on a separate Customer Information Directory located on accounting database 144. As shown in Figure 4, the Customer Information Directory Table on accounting database 144 is formed of a plurality of individual records 450, each of which contains several fields for storing information about a particular user. Fields 452, 454, 456, 458 and 460 store substantially the same information as that which is stored respectively in fields 402, 404, 406, 408 and 410, respectively, described above. However, in step 240 the user is also prompted by the enrollment means 145 to choose an enrollment plan and enter certain personal financial information which is then stored in records 450. In a preferred embodiment, the user may select either a free trial membership or one of several active

membership plans, and a code representing the enrollment plan selected by the user is then stored in field 462. If the user has selected either an active or free trial membership plan, the user is prompted to enter credit card information for paying for the selected enrollment plan. This credit card information is stored in field 474 and used by billing means 143 to verify that the user is credit worthy.

Referring again to Figure 2, following the entry of the user information into records 400, 450, processing proceeds to step 250, where enrollment means 145 downloads a copy of user software 106a onto user station 102a. Unlike the user software 106 described above, user software 106a does not include any embedded CM ID information, or, to state it another way, user software 106a contains a "null" CM ID field embedded therein.

Referring still to Figure 2, if a determination is made in step 220 that the user is operating on a user station 102 that connected to OLS 140 through channel 108, then processing proceeds to step 260 where enrollment means 145 determines a CM ID associated with the user station 102a. In contrast to step 230, the CM ID is determined in step 260 from an embedded CM ID stored on user software 106 which was previously loaded (in step 202) on user station 102. Next, in step 270, enrollment means 145 attempts to enroll the user in OLS 140. Step 270 is substantially the same as step 240 described above, except in step 270 the CM ID used for creating and updating the records 400 and 450 represents the CM ID embedded in software 106, as opposed to a CM ID determined from a UNIX symbolic link passed to OLS 140 over WWW 120.

Following either step 250 or 270, processing proceeds to step 280, where enrollment means 145 communicates with billing means 143 to determine whether the user is credit worthy. In addition, enrollment means 145 determines (based on the information stored in field 462) whether the user has enrolled as an active (i.e., non-trial) user. If the user is credit worthy and has enrolled as an active user, processing proceeds to step 285, where a payment record for paying a one-time bounty (or referral fee) to the co-marketer that directed the user to OLS 140 is created.

As explained more fully below, the amount of the one-time bounty payment created in step 285 is preferably dependent on the number of users previously directed to OLS 140 by the co-marketer during a previous period (month or quarter). A Co-Marketer Information Directory located on accounting database 144 is provided for storing information about each authorized co-marketer on OLS 140. As shown in Figure 4, the Co-

Marketer Information Directory Table on accounting database 144 is formed of a plurality of records 480, each of which contains several fields for storing information about a particular authorized co-marketer. More particularly, a field 482 is provided for storing the CM ID associated with a co-marketer, and fields 484, 486 and 488 are respectively provided for storing name, address and telephone number information representing the co-marketer. In addition, each time a user is enrolled on OLS 140, the values in fields 492 and 496, which respectively represent the total number of users directed to OLS 140 by the co-marketer and the number of users directed to OLS 140 during the current month are incremented. Each record 480 also contains a field 494 representing the number of users that were directed to OLS by the co-marketer during the previous month. In a preferred embodiment of the present invention, the value stored in field 494 is used in step 285 in calculating the amount of the one time bounty payment to be paid to the co-marketer. More particularly, a higher one-time bounty payment will be paid to a co-marketer in step 285 if the number of prior enrollees represented in field 494 exceeds a predetermined threshold.

When a user reaches OLS site 128 from a previous location on WWW 120, the user will typically be initially directed to the home page at web site 128. In a first embodiment of the present invention described above in connection with Figure 2, the user may enroll on OLS 140 directly from the home page of site 128 upon reaching site 128. In alternate preferred embodiments described below, the user may browse through the home web page of site 128, and then through various further web pages at site 128, prior to reaching an enrollment page at site 128 (e.g., WWW.OLS.COM\...\ENROLL\ENROLL.P1) from which the user then enrolls onto OLS 140. These alternate preferred embodiments are described respectively in connection with Figures 5 and 6. Since the systems described in connection with Figures 5 and 6 permit a user to traverse multiple pages at site 128 prior to enrolling on OLS 140, these systems function to preserve the UNIX symbolic link information originally passed to OLS site 128 from a prior web site as the user moves between web pages at site 128.

Referring now to Figure 5, there is shown a schematic diagram illustrating the use of UNIX symbolic links and relative URL addressing for moving between page locations at OLS site 128, in accordance with a preferred embodiment of the present invention. URL 502 points to the home page address of OLS site 128 on the WWW. URLs 504, 506, 508 also point to the home page address of OLS site 128 on the WWW; however, URLs 504,

506, 508 each include a UNIX symbolic link (/CM1, /CM2, /CM3) appended thereto. As described above, in the present invention, each UNIX symbolic link appended to a URL represents the identity of a co-marketer that directed a user to the home page address of OLS site 128. When the user arrives at the home page of OLS site 128, the user may then browse through various pages provided by OLS site 128 on the WWW. For example, the user may view pages providing information about OLS 140 by clicking on a "table of contents" entry on the home page of OLS site 128. Upon clicking on this "table of contents" entry on the home page, the user is directed to a Table of Contents Page represented by URL 510. From this Table of Contents Page, the user may click on individual pages (e.g., Info.P1, Info.P2, etc.) listed on the Table of Contents Page. Upon clicking on an individual page such as, for example, Information Page 1, the user is directed to a first information page represented by URL 514.

As described above in the background section, when relative URL addressing is used to move between pages on WWW 120, a user may only move between pages in the user's current directory or to a subdirectory located below the user's current directory in a directory tree such as that shown in Figure 5. Thus, when standard relative URL addressing is used, it is not possible for a user to move from the page represented by URL 514 to the page represented by URL 518 and still preserve the UNIX symbolic link/CMID information described above. In the example shown in Figure 5, the page 514a represented by URL 514 contains a box giving the user an option to enroll on OLS 140. In accordance with the present invention, if the user clicks on the "Enroll on OLS" box on page 514a, a special redirecting program (redirect.cgi) is triggered on web server 142 for redirecting the user from the page represented by URL 514 to the OLS enrollment page represented by URL 518. A pseudo-code version of the redirect.cgi program is shown in Table I below:

```
main( input_parameters )
{       // last_url will hold the URL for the page the user was on
        // when they wanted to redirect upward. (for example
        // "http://www.ols.com/cm1/subdir1/subdir2/subdir3/info2.html")

        last_url = input_parameter[x];

        // destination will hold the page they want to redirect to.
        // It contains one or more "../" substrings.
        // (example. "../../subdirx/enroll.htm")
```

```
destination = input_parameter[y];

// count_substrings() counts the number of substrings ("../")
// within the given string of characters (destination).
// In this example it would return the number "2".

number_of_levels_up = count_substrings("../", destination ) ;

// remove_n_levels() takes a fully specified URL such as
// "http://www.ols.com/cm1/subdir1/subdir2/subdir3/page.htm"
// and removes a given number of directories and the page name.
// For example if number_of_levels_up is 2, the output from
// remove_n_levels(last_url,2) would be
// "http://www.ols.com/cm1/subdir1/"

new_directory = remove_n_levels( last_url, number_of_levels_up );

// get_relative_url() takes the given string and returns the portion
// after all of the "../" substrings. Thus in this example it returns
// "subdirx/enroll.htm"

relative_url = get_relative_url( destination );

// concatenate() takes 2 strings and splices the second one
// onto the back of the first. (in the example this yields:
// "http://www.ols.com/cm1/subdir1/subdirx/enroll.htm")

new_absolute_url = concatenate( new_directory, relative_url );

// redirect_browser() sends a "Hyper-Text Transfer Protocol (http)"
// message back to the user's web browser telling it to get the given URL.

redirect_browser( new_absolute_url );
}
```

TABLE I

Thus, the redirect.cgi program accepts as arguments the current URL of the user (e.g., URL 514) and a destination URL representing the location to which the user desires to move (e.g., URL 518). The program then strips the ".../Info/Info.P1" portion off of the current URL 514, and replaces the striped portion with the ".../Enroll/Enroll.P1" portion of destination URL 518 to form a new URL which is then used for redirecting the user to the page represented by URL 518. The redirect.cgi program is significant to the operation of the present invention because, among other things, this program allows the UNIX symbolic link information that was originally passed when the user arrived at the home page of OLS site 128

to be retained as the user moves between pages at OLS site 128. Thus, the redirecting.cgi program insures that the UNIX symbolic link information provided by a co-marketer will be present when the enrollment means 145 attempts to enroll the user on OLS 140.

The redirect.cgi program discussed in connection with Figure 5 and Table I above represents a first preferred system for retaining at site 128 the UNIX symbolic link information (that was originally passed when the user originally arrived at OLS site 128 from a previous site) as the user moves between web pages at OLS site 128. In accordance with an alternative preferred embodiment of the present invention, a further system (described in connection with Figure 6 and Table II below) may alternatively be used to store and transmit the UNIX symbolic link information that was originally passed when the user arrived at the home page of OLS site 128. In this alternate embodiment, the URL used to direct a user from a previous site (e.g., 122a, 122b, 122c) to OLS site 128 includes a string which functions to call a special page_link.cgi program which runs on web server 142. The string passed to OLS site 128 also contains (i) a destination page identifier (or filename) representing the particular web page at site 128 to which the user has been directed by the previous web site, and (ii) a UNIX symbolic link or CMID code associated with the previous web site. More particularly, the destination page identifier and the UNIX symbol link information/CMID code are included in the string as arguments to the page_link.cgi program. An exemplary URL which invokes the page_link.cgi program and that could be used by co-marketer site 122a for directing a web user from a site 122a to the home page of site 128 is shown below:

WWW.OLS.COM\page_link.cgi ? index @ CM1

The first portion (i.e., WWW.OLS.COM) of this exemplary URL identifies web site 128 as the web site to which the user is being directed. The remaining portion (i.e., page_link.cgi ? index @ CM1) of the URL represents a call to the page_link.cgi program. The program call includes two arguments, namely, a destination page identifier (i.e., index) representing the particular page at site 128 to which the user has been directed, and a UNIX symbolic link/CMID code (i.e., CM1) representing the identity of the web site 122a that directed the user to site 128.

Referring now to Figure 6, there is shown a flow diagram of a system 600 for implementing the page_link.cgi program. In step 610, when web server 142 receives a URL

which includes a string containing a call to the page_link.cgi program, the page_link.cgi program is invoked on the web server 142. Next, in step 620, the page_link.cgi program extracts the destination page identifier (e.g., index) and UNIX symbolic link/CMID code (e.g., CM1) that were contained as arguments in the page_link.cgi program call. Next, the page at web site 128 represented by the destination page identifier is retrieved. Each page at web site 128 is represented by a file which includes one or more fields containing further URLs representing links to other pages at web site 128 (internal page links) or to pages at web sites other than site 128 (external page links). Each URL in the destination page is then selected and tested (in steps 630 and 640) in order to determine whether the URL includes a string for calling the page_link.cgi program. If the URL does include the "page_link.cgi" string, a further determination is made (in step 650) whether the URL represents an internal or external page link. In this step, the URL will be determined as representing an internal page link if the first portion of the URL represents OLS site 128 (i.e., "WWW.OLS.COM") or if there is no site name portion in the URL; otherwise the URL will be determined as representing an external page link. Next, in step 660, for each internal URL in the destination page which includes a string for calling the page_link.cgi program, the page_link.cgi program appends the UNIX symbolic link/CMID code (i.e., CM1) originally passed as an argument to the program to the end of the URL. In addition, in step 670, for each external URL in the destination page which includes the "page_link.cgi" string, the page_link.cgi program appends the UNIX symbolic link/CMID originally passed to the program followed by a UNIX symbolic link/CMID representing OLS site 128 (e.g., /OLS) to the end of the URL. The process is then repeated from step 630 for each URL on the destination page. The destination page, which includes URLs having the appended codes described above is then passed back to the user in step 690. Thereafter, when the user desires to move off of the destination page (passed to the user in step 690), the user will select one of the URL page links on the user's page as a new destination page. If the URL corresponding to this new destination page contains a call to the page_link.cgi program described above, the process described above is repeated from step 610 using the URL of the new destination. A pseudo-code listing of an exemplary web page file and of the page_link.cgi program are shown below in Table II:

** WEB PAGE FILE PSEUDO CODE**

<HTML>

```
<HEAD>
<TITLE>Online Service Home Page</TITLE>
</HEAD>
<BODY>
<A HREF="http://www.ols.com/cgi-bin/page_link.cgi?enroll&CM1">Enroll page on OLS 140</A>

<A HREF="http://www.other.com/">Visit another company's web pages</A>

<A HREF="http://www.cm.com/ cgi-bin/page_link.cgi?index&CM1&OLS">Maintain comarketer info and visit another site</A>
</BODY>
</HTML>

** PAGE LINK CGI PSUEDO CODE **

PageLink (URL)  {
        get destinationPageName and CMID_string from URL

        // assuming the following URL:
        // "http://www.ols.com/cgi-bin/page_link.cgi?index&CM1"
        // destinationPageName is index, as it follows "page_link.cgi?"
        // and the CMID_string is "CM1", as it follows the
        // destinationPageName (and is separated by a "&").
        //
        // CMID_string consists of one or more comarketer codes
        // separated by "&" characters, and records the path taken
        // by the user through web sites which employ this tracking
        // system

        put contents of page named destinationPageName into destinationPage

        for each URL in destinationPage
        {
                if URL contains page_link.cgi call and refers to an internal URL
                {
                        // if there is no site name or if the site name matches
                        // this on-line service's, the URL is internal

                        append CMID_string to URL
                }
                else if URL contains page_link.cgi call and is an external URL
                {
                        // if there is a site name and it does not match that of OLS 140,
                        // the URL is external

                        append CMID_string to URL
                        append OLS_string to URL
```

```
                    // OLS_string refers to the CMID code of
                    // the site running this instance of the page_link.cgi
                    // application, and is unique among all sites participating.
                }

                // note that nothing is done to URLs which do not refer to
                // the page_link.cgi application, since no comarketer code
                // tracking is done when users follow these links
            }
            send page
        }
```

TABLE II

In the embodiment shown in Figure 6 and described above, each page link URL on a web page at site 128 will preferably include a call to the page_link.cgi program if the page link points to either (i) a further web page at site 128, or (ii) a further web site which is adapted to recognize UNIX symbolic links that have been inserted into a URL by a previous web site during a user session. By inserting the page_link.cgi program call and UNIX symbolic link information into each page link that points to a further web page at site 128, the system insures that the UNIX symbolic link information originally passed to site 128 by a previous web site will be available when OLS 140 attempts to enroll the user into OLS 140. In addition, by inserting the UNIX symbolic link information associated with both a previous web site 122a, b, c and OLS site 140 into page links associated with different web sites (other than site 128), the system permits the user to carry UNIX symbolic link information representing previous location(s) traversed during a user session to further web sites.

Referring now to Figure 7, there is shown a flow diagram illustrating the operation of a system 700 for generating recurring bounty payment records, in accordance with a preferred embodiment of the present invention. In addition to providing each co-marketer with a one-time bounty payment each time a user directed to OLS 140 by the co-marketer enrolls on OLS 140, billing means 143 also generates "recurring" bounty payment records for certain selected co-marketers that have secured a preferred status with OLS 140. Information representing whether or not a particular co-marketer has such preferred status (and is thus eligible to receive a recurring bounty payment) is stored in field 490 of records 480. As shown in system 700, a recurring bounty payment is determined for each preferred co-marketer on a periodic basis based on the number of users that were referred to OLS 140

by the co-marketer (at any time) and which are still active subscribers on OLS 140.

Referring still to Figure 7, in step 710 a counter used for determining the total number of active users associated with each co-marketer is initialized, preferably to zero. Next, in step 720 a co-marketer record 480 having a preferred status (as indicated by field 490) is selected for processing. Next, a user record 450 having the same CM ID (as indicated by field 454) as that of the selected co-marketer is selected for processing. If field 470 of the selected user record indicates that the selected user is still an active user and the user has been an active user on OLS 140 for at least 90 days (as indicated by field 476), then processing proceeds to step 750 where the recurring user counter is incremented. Next, in step 760, the process is repeated from step 730 until each user record 450 having the same CM ID as the selected co-marketer has been processed. Thereafter, in step 770, the billing means 143 generates a recurring bounty payment for the selected co-marketer by multiplying a value represented by the recurring user counter with a per user bounty amount. Finally, in step 780, the process is repeated from step 710 until each co-marketer having a preferred status has been processed.

In an alternate preferred embodiment of the present invention (not shown), the value stored in field 492 may be used by system 700 (at step 770) in calculating the rate of each recurring bounty payment to be paid to each preferred co-marketer. More particularly, a higher recurring bounty payment rate may be paid to a co-marketer if the value of enrollees represented in field 492 exceeds a predetermined threshold.

Furthermore, it is to be understood that although the present invention has been described with reference to a preferred embodiment, various modifications, known to those skilled in the art, may be made to the structures and process steps presented herein without departing from the invention as recited in the several claims appended hereto.

What is claimed is:

1. A method for redirecting a user from a first location on a world wide web (WWW) to a second location on said WWW, wherein relative universal resource locator (URL) addressing is used during said redirecting process, comprising the steps of:

(A) receiving a signal from said first location indicating that said user wishes to move from said first location on said WWW to said second location on said WWW;

(B) passing, in response to said signal, a current URL representing an address of said first location on said WWW and a destination URL portion representative of an address of said second location on said WWW to a redirecting means, said current URL having first and second portions;

(C) forming a destination URL, with said redirecting means, by substituting said destination URL portion in place of said second portion in said current URL, wherein said destination URL represents a relative address of said second location on said WWW; and

(D) moving said user from said first location on said WWW to said second location on said WWW in accordance with said destination URL formed in step (C).

2. The method of claim 1, wherein said destination URL portion recited in step (B) is formed from a directory identifier associated with said second location.

3. The method of claim 2, wherein said first portion of said current URL recited in step (B) is formed from a web page address identifier associated with said first and second locations.

4. The method of claim 3, wherein said first portion of said current URL recited in step (B) is further formed of a UNIX symbolic link representing a prior WWW location traversed by said user before said user reached said first location on said WWW.

5. The method of claim 4, wherein said prior WWW location represents a co-marketer of on-line services on said WWW and said web page identifier is associated with an on-line service on said WWW.

6. The method of claim 5, further comprising the step of:

(E) using, at said second location, said UNIX symbolic link to pay a bounty

to said co-marketer.

7. An apparatus for redirecting a user from a first location on a world wide web (WWW) to a second location on said WWW, wherein relative universal resource locator (URL) addressing is used during said redirecting, comprising:

(A) means for receiving a signal from said first location indicating that said user wishes to move from said first location on said WWW to said second location on said WWW;

(B) means for passing, in response to said signal, a current URL representing an address of said first location on said WWW and a destination URL portion representative of an address of said second location on said WWW to a redirecting means, said current URL having first and second portions;

(C) said redirecting means including means for forming a destination URL by substituting said destination URL portion in place of said second portion in said current URL, wherein said destination URL represents a relative address of said second location on said WWW; and

(D) means for moving said user from said first location on said WWW to said second location on said WWW in accordance with said destination URL formed by said redirecting means.

8. The apparatus of claim 7, wherein said destination URL portion is formed from a directory identifier associated with said second location.

9. The apparatus of claim 8, wherein said first portion of said current URL is formed from a web page address identifier associated with said first and second locations.

10. The apparatus of claim 9, wherein said first portion of said current URL is further formed of a UNIX symbolic link representing a prior WWW location traversed by said user before said user reached said first location on said WWW.

11. The apparatus of claim 10, wherein said prior WWW location represents a co-marketer of on-line services on said WWW and said web page identifier is associated with an on-line service on said WWW.

12. The apparatus of claim 11, further comprising:

(E) bounty payment means, responsive to said UNIX symbolic link, for paying a bounty to said co-marketer.

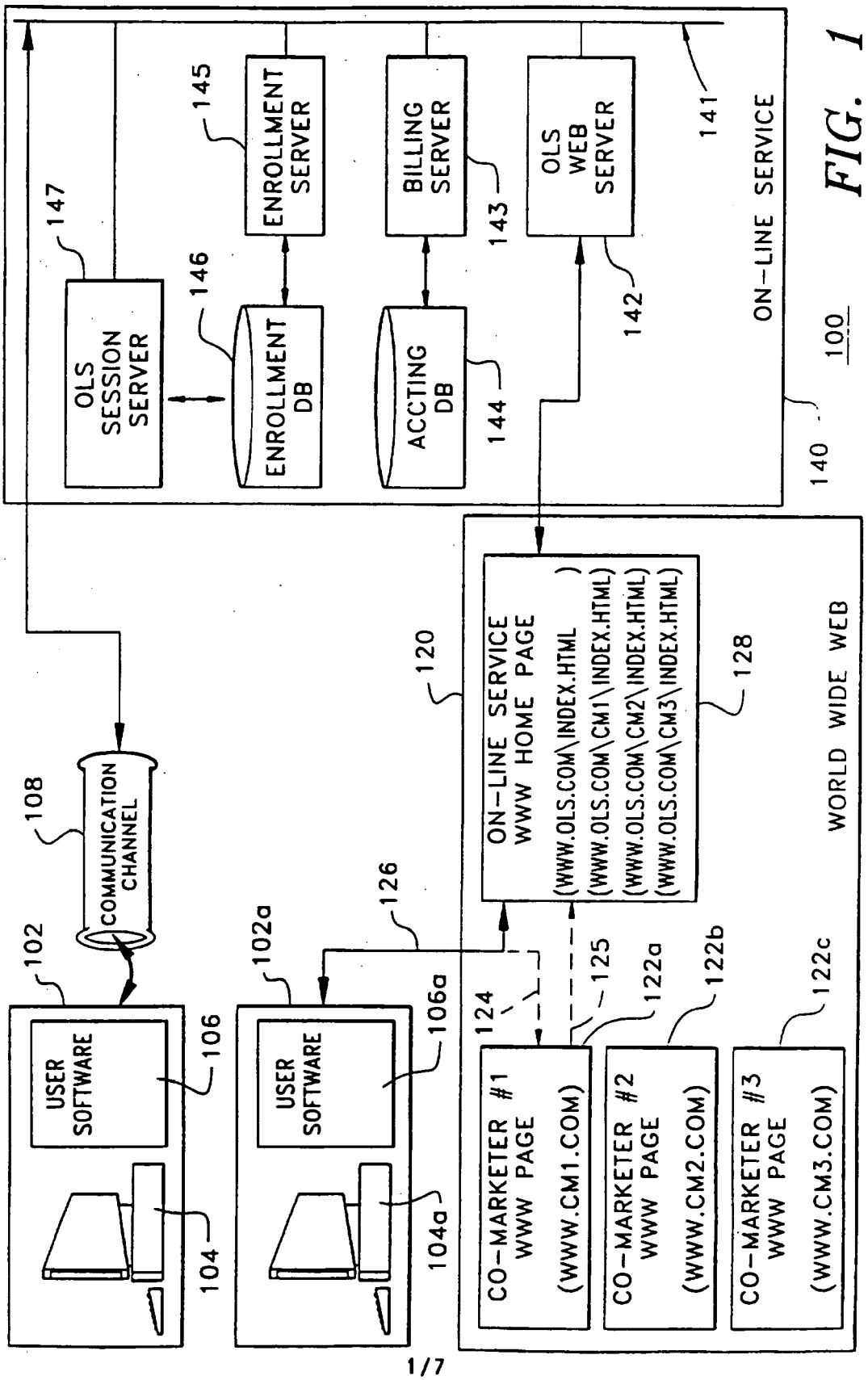

ON-LINE SERVICE
100
140
OLS SESSION SERVER
147
ENROLLMENT DB
146
ENROLLMENT SERVER
145
ACCTING DB
144
BILLING SERVER
143
OLS WEB SERVER
142
141
COMMUNICATION CHANNEL
108
ON-LINE SERVICE WWW HOME PAGE
(WWW.OLS.COM\INDEX.HTML)
(WWW.OLS.COM\CM1\INDEX.HTML)
(WWW.OLS.COM\CM2\INDEX.HTML)
(WWW.OLS.COM\CM3\INDEX.HTML)
120
128
WORLD WIDE WEB
102
USER SOFTWARE
106
104
102a
USER SOFTWARE
106a
104a
126
124
125
CO-MARKETER #1 WWW PAGE
(WWW.CM1.COM)
122a
CO-MARKETER #2 WWW PAGE
(WWW.CM2.COM)
122b
CO-MARKETER #3 WWW PAGE
(WWW.CM3.COM)
122c

FIG. 1

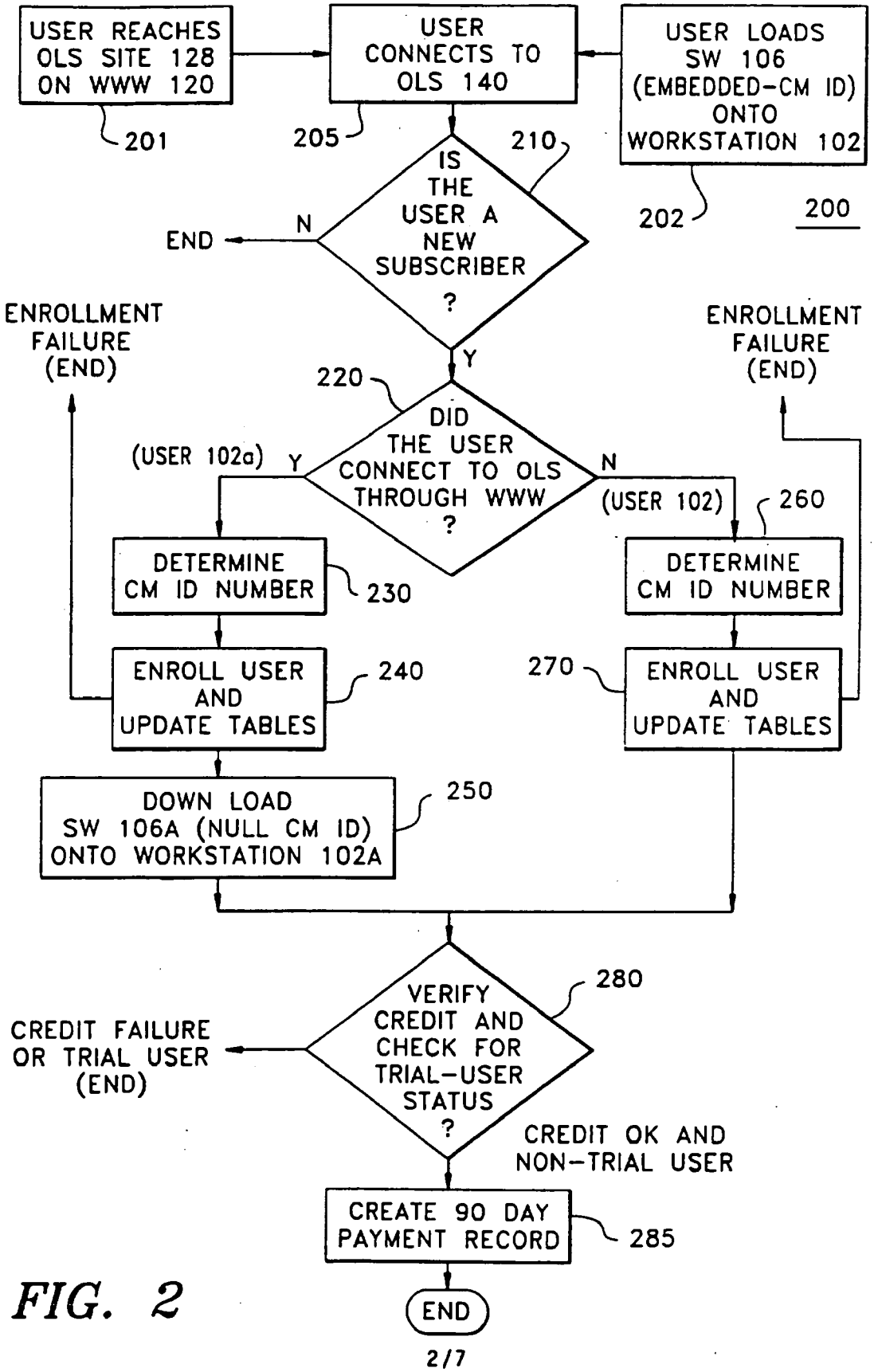

USER REACHES OLS SITE 128 ON WWW 120
201
USER CONNECTS TO OLS 140
205
USER LOADS SW 106 (EMBEDDED-CM ID) ONTO WORKSTATION 102
202
200
IS THE USER A NEW SUBSCRIBER ?
210
N
END
Y
ENROLLMENT FAILURE (END)
ENROLLMENT FAILURE (END)
220
DID THE USER CONNECT TO OLS THROUGH WWW ?
(USER 102a)
Y
N
(USER 102)
260
DETERMINE CM ID NUMBER
230
DETERMINE CM ID NUMBER
ENROLL USER AND UPDATE TABLES
240
270
ENROLL USER AND UPDATE TABLES
DOWN LOAD SW 106A (NULL CM ID) ONTO WORKSTATION 102A
250
VERIFY CREDIT AND CHECK FOR TRIAL-USER STATUS ?
280
CREDIT FAILURE OR TRIAL USER (END)
CREDIT OK AND NON-TRIAL USER
CREATE 90 DAY PAYMENT RECORD
285
END

FIG. 2

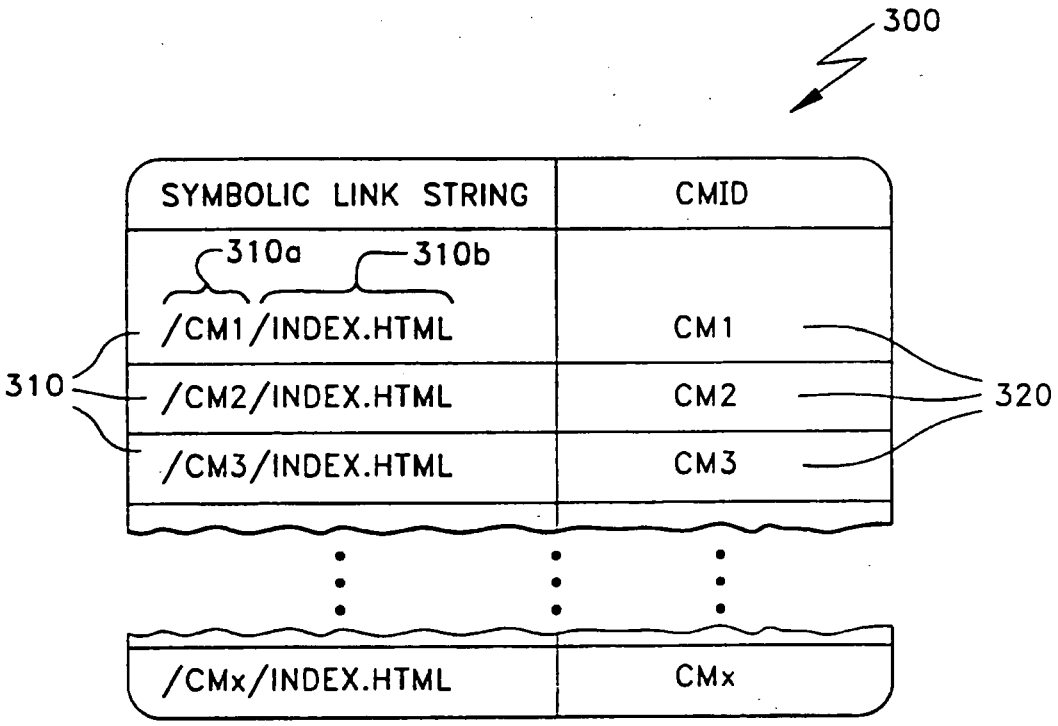

300
SYMBOLIC LINK STRING
CMID
310a
310b
/CM1/INDEX.HTML
CM1
310
/CM2/INDEX.HTML
CM2
320
/CM3/INDEX.HTML
CM3
/CMx/INDEX.HTML
CMx

FIG. 3

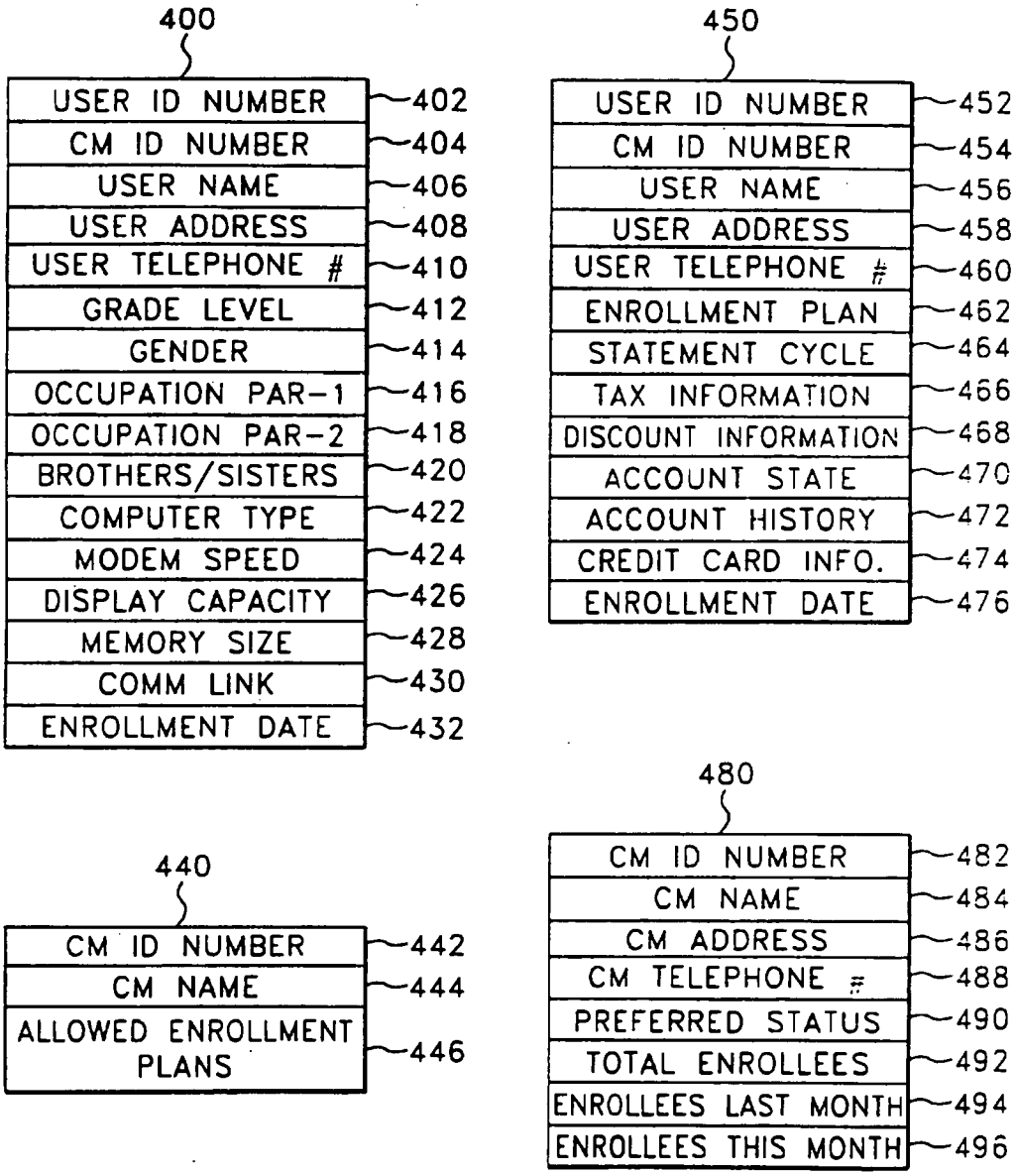

400
USER ID NUMBER 402
CM ID NUMBER 404
USER NAME 406
USER ADDRESS 408
USER TELEPHONE # 410
GRADE LEVEL 412
GENDER 414
OCCUPATION PAR-1 416
OCCUPATION PAR-2 418
BROTHERS/SISTERS 420
COMPUTER TYPE 422
MODEM SPEED 424
DISPLAY CAPACITY 426
MEMORY SIZE 428
COMM LINK 430
ENROLLMENT DATE 432
450
USER ID NUMBER 452
CM ID NUMBER 454
USER NAME 456
USER ADDRESS 458
USER TELEPHONE # 460
ENROLLMENT PLAN 462
STATEMENT CYCLE 464
TAX INFORMATION 466
DISCOUNT INFORMATION 468
ACCOUNT STATE 470
ACCOUNT HISTORY 472
CREDIT CARD INFO. 474
ENROLLMENT DATE 476
440
CM ID NUMBER 442
CM NAME 444
ALLOWED ENROLLMENT PLANS 446
480
CM ID NUMBER 482
CM NAME 484
CM ADDRESS 486
CM TELEPHONE # 488
PREFERRED STATUS 490
TOTAL ENROLLEES 492
ENROLLEES LAST MONTH 494
ENROLLEES THIS MONTH 496

FIG. 4

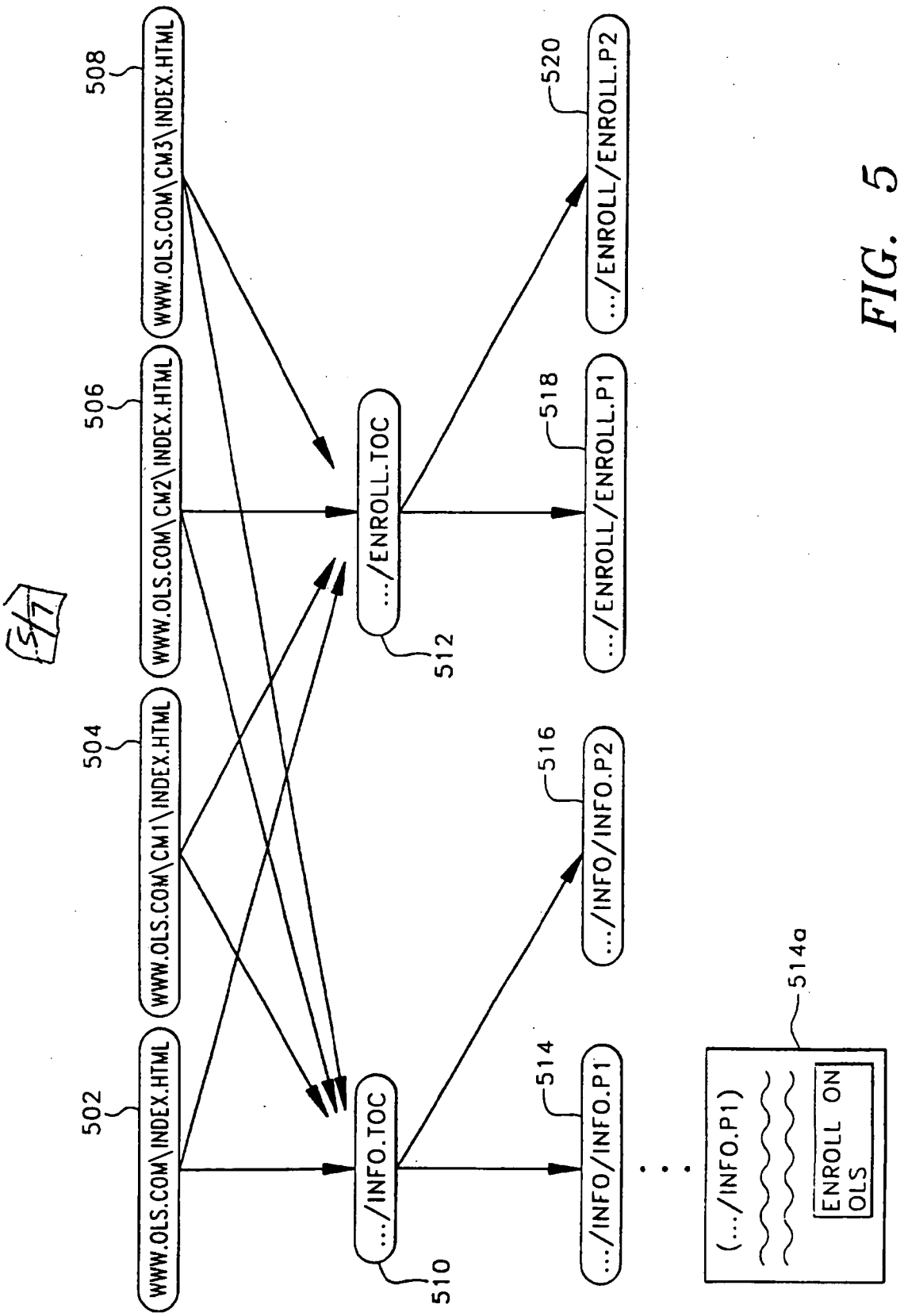

502
WWW.OLS.COM\INDEX.HTML
504
WWW.OLS.COM\CM1\INDEX.HTML
506
WWW.OLS.COM\CM2\INDEX.HTML
508
WWW.OLS.COM\CM3\INDEX.HTML
510
.../INFO.TOC
512
.../ENROLL.TOC
514
.../INFO/INFO.P1
516
.../INFO/INFO.P2
518
.../ENROLL/ENROLL.P1
520
.../ENROLL/ENROLL.P2
514a
(.../INFO.P1)
ENROLL ON OLS

FIG. 5

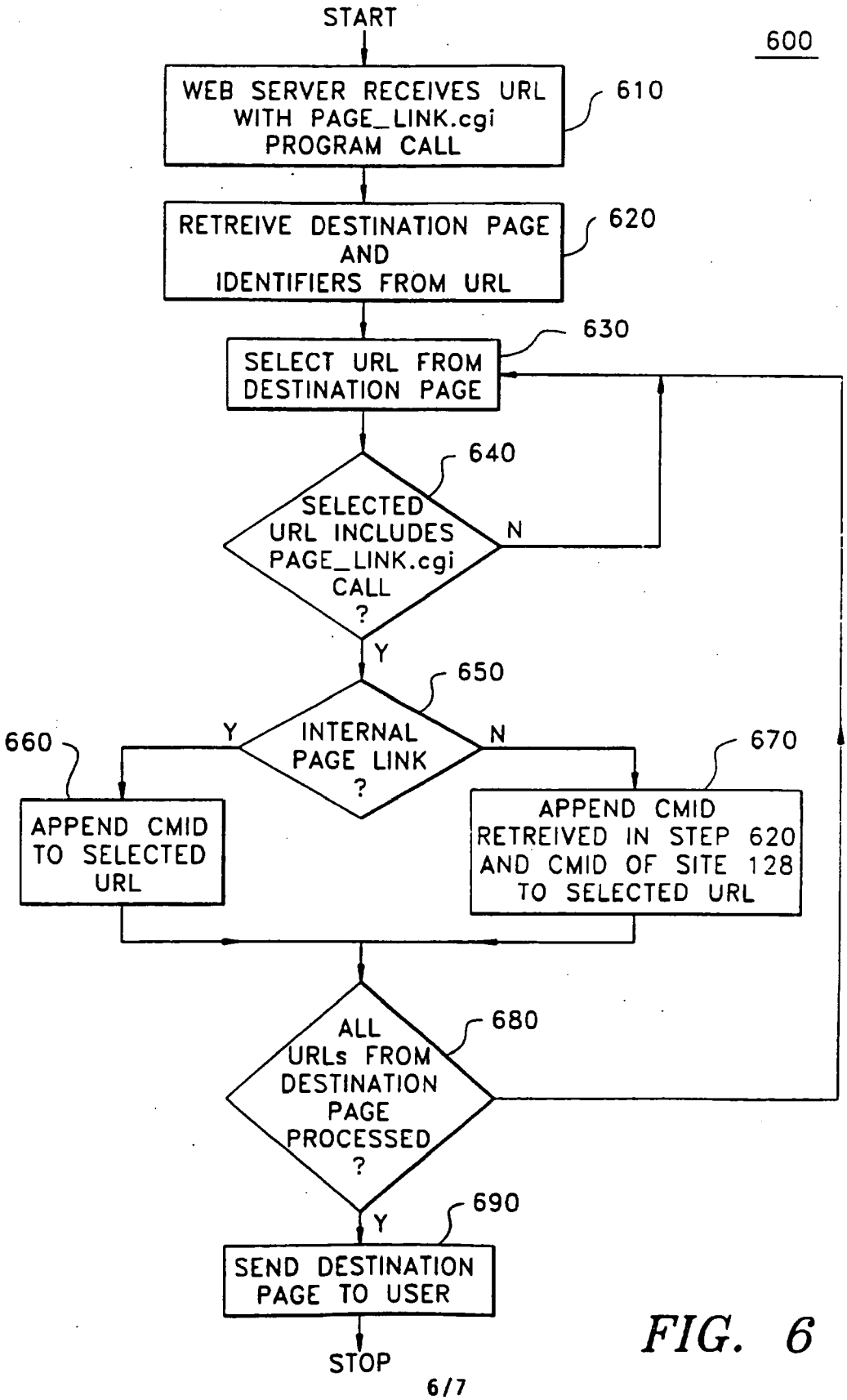

START
600
WEB SERVER RECEIVES URL WITH PAGE_LINK.cgi PROGRAM CALL
610
RETREIVE DESTINATION PAGE AND IDENTIFIERS FROM URL
620
SELECT URL FROM DESTINATION PAGE
630
SELECTED URL INCLUDES PAGE_LINK.cgi CALL ?
640
N
Y
INTERNAL PAGE LINK ?
650
Y
N
660
APPEND CMID TO SELECTED URL
670
APPEND CMID RETREIVED IN STEP 620 AND CMID OF SITE 128 TO SELECTED URL
ALL URLs FROM DESTINATION PAGE PROCESSED ?
680
Y
690
SEND DESTINATION PAGE TO USER
STOP

FIG. 6

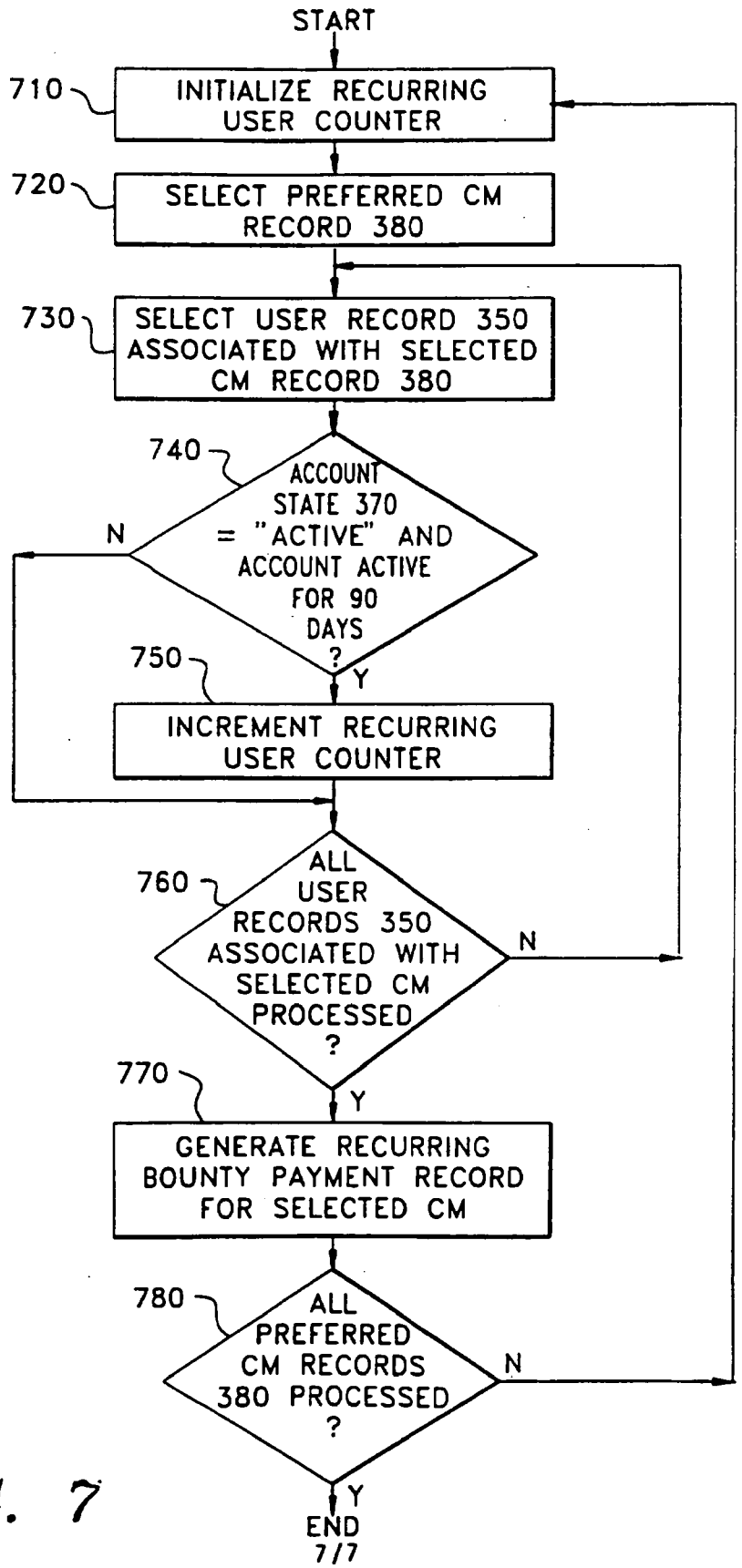

START
700
710
INITIALIZE RECURRING USER COUNTER
720
SELECT PREFERRED CM RECORD 380
730
SELECT USER RECORD 350 ASSOCIATED WITH SELECTED CM RECORD 380
740
ACCOUNT STATE 370 = "ACTIVE" AND ACCOUNT ACTIVE FOR 90 DAYS ?
N
Y
750
INCREMENT RECURRING USER COUNTER
760
ALL USER RECORDS 350 ASSOCIATED WITH SELECTED CM PROCESSED ?
N
Y
770
GENERATE RECURRING BOUNTY PAYMENT RECORD FOR SELECTED CM
780
ALL PREFERRED CM RECORDS 380 PROCESSED ?
N
Y
END

FIG. 7

7/7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US96/14988

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 13/14, 13/42; H04L 12/46, 29/02
US CL :395/200.09, 200.12
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/200.09, 200.12, 200.15; 380/25, 49; 340/825.25

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

NONE

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

USPTO APS
search terms: HTML, HTTP, hypertext, hypermedia, WWW, internet, URL, access path, trusted path, world wide web

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 5,355,472 A (LEWIS) 11 October 1994 Abstract; Fig. 3-5; Col. 1, lines 31-63	1-12
A, P	US 5,537,546 A (SAUTER) 16 July 1996 Abstract; Fig. 1; Col. 3, lines 1-45	1-12
A, P	US 5,544,322 A (CHENG et al.) 06 August 1996 Abstract; Fig. 3; Col. 3, lines 20-60	1-12
A, P	US 5,561,706 (FENNER) 01 October 1996 Abstract; Fig. 1-4; Col. 1, lines 23-62	1-12

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be part of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search 27 NOVEMBER 1996	Date of mailing of the international search report 26 DEC 1996
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer MARK H. RINEHART BRIAN A. HARDEN PARALEGAL SPECIALIST GROUP 2400 Telephone No. (703) 305-9600

Form PCT/ISA/210 (second sheet)(July 1992)*

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/46, 17/30, H04L 29/06 — A1

(11) International Publication Number: **WO 97/29423**

(43) International Publication Date: 14 August 1997 (14.08.97)

(21) International Application Number: PCT/GB96/02983

(22) International Filing Date: 4 December 1996 (04.12.96)

(30) Priority Data:
9601559.9 26 January 1996 (26.01.96) GB

(71) Applicant: INTERNATIONAL BUSINESS MACHINES CORPORATION [US/US]; Armonk, NY 10504 (US).

(71) Applicant *(for MC only)*: IBM UNITED KINGDOM LIMITED [GB/GB]; P.O. Box 41, North Harbour, Portsmouth, Hampshire PO6 3AU (GB).

(72) Inventors: STANFORD-CLARK, Andrew, James; 68 Allbrook Knoll, Allbrook, Eastleigh, Hampshire SO50 4RY (GB). MARTIN, Sean, James; 268B West End Lane, West Hampstead, London NW6 1LJ (GB).

(74) Agent: JENNINGS, Michael, John; IBM United Kingdom Limited, Intellectual Property Dept., Hursley Park, Winchester, Hampshire SO21 2JN (GB).

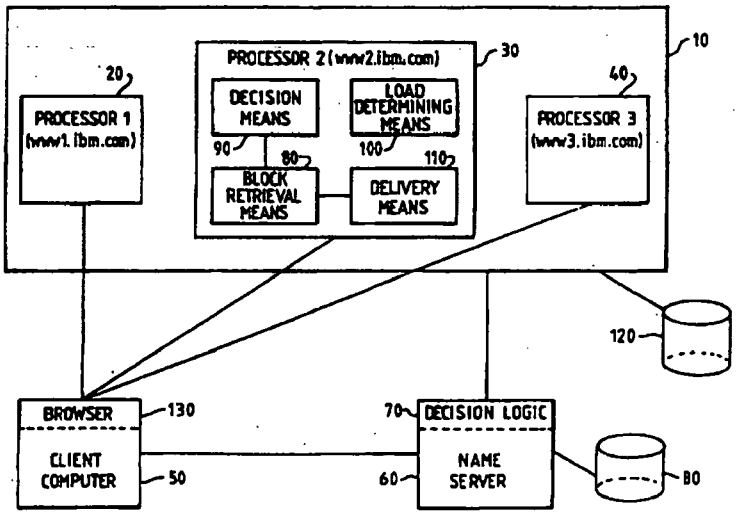
(81) Designated States: BR, CA, CN, CZ, HU, JP, KR, PL, RU, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: LOAD BALANCING ACROSS THE PROCESSORS OF A SERVER COMPUTER

(57) Abstract

The present invention relates to a server computer connectable to a network and having a plurality of processors (20, 30, 40) arranged to provide a service to one or more client computers connected to the network. The service involves the provision of blocks of information to a client computer (50), and the processors have access to a data storage (120) in which data for the blocks of information is stored. According to preferred embodiments of the invention, each processor in the server has block retrieval means (80), employed in response to a request from a client computer (50) for a particular block of information from that processor, to retrieve data from the data storage (120) in order to construct a file representing the requested block of information, the file including references to subsequent blocks of information requestable by the client (50). Additionally, load determining means (100) is provided of periodically determining activity data for the processor for inclusion in a load distribution record maintained for all of the processors of the server. Decision means (90) is then used which, for each reference to a subsequent block of information in the file constructed by the block retrieval means (80), is arranged to determine, based on the load distribution record, which processor should service a request from the client computer (50) for that subsequent block of information, and include an address for that processor in the file constructed by the block retrieval means (80). Once the decision means has completed this process, the file is then sent to the client computer (50). In preferred embodiments, the service being provided is the World Wide Web (WWW) service, and the technique is employed on a parallel Web server providing Web pages to clients over the Internet. By employing the technique of the invention, more effective load balancing across the processors of the Web server is possible, thereby enabling problems such as "toasting" of a particular processor to be averted.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AM	Armenia	GB	United Kingdom	MW	Malawi
AT	Austria	GE	Georgia	MX	Mexico
AU	Australia	GN	Guinea	NE	Niger
BB	Barbados	GR	Greece	NL	Netherlands
BE	Belgium	HU	Hungary	NO	Norway
BF	Burkina Faso	IE	Ireland	NZ	New Zealand
BG	Bulgaria	IT	Italy	PL	Poland
BJ	Benin	JP	Japan	PT	Portugal
BR	Brazil	KE	Kenya	RO	Romania
BY	Belarus	KG	Kyrgystan	RU	Russian Federation
CA	Canada	KP	Democratic People's Republic of Korea	SD	Sudan
CF	Central African Republic			SE	Sweden
CG	Congo	KR	Republic of Korea	SG	Singapore
CH	Switzerland	KZ	Kazakhstan	SI	Slovenia
CI	Côte d'Ivoire	LI	Liechtenstein	SK	Slovakia
CM	Cameroon	LK	Sri Lanka	SN	Senegal
CN	China	LR	Liberia	SZ	Swaziland
CS	Czechoslovakia	LT	Lithuania	TD	Chad
CZ	Czech Republic	LU	Luxembourg	TG	Togo
DE	Germany	LV	Latvia	TJ	Tajikistan
DK	Denmark	MC	Monaco	TT	Trinidad and Tobago
EE	Estonia	MD	Republic of Moldova	UA	Ukraine
ES	Spain	MG	Madagascar	UG	Uganda
FI	Finland	ML	Mali	US	United States of America
FR	France	MN	Mongolia	UZ	Uzbekistan
GA	Gabon	MR	Mauritania	VN	Viet Nam

LOAD BALANCING ACROSS THE PROCESSORS OF A SERVER COMPUTER

Field of the Invention

The present invention relates to server computers connectable to a network for providing a service to one or more client computers also connected to the network. In particular it relates to server computers of the type having a plurality of processors arranged to provide the service, and concerns the problem of balancing the load on those processors resulting from requests from the client computers.

Background of the Invention

There is increasing demand for high-powered server computers for networks in general, and in particular for the Internet. The Internet is a giant network made up of many smaller networks interlinked through the use of devices such as bridges and gateways. In theory, any computer connected to the Internet is capable of exchanging information with any other computer that is connected to the Internet. However, since computers often contain sensitive information, many computers and networks have security schemes that restrict the exchange of information among computers to various degrees.

Since the Internet represents a very powerful structure through which to disseminate information, a number of services have evolved which make use of the Internet to pass information from one computer to another. Examples of these services are FTP, News, Gopher, and the World Wide Web (or WWW as it will be referred to hereafter).

The WWW is one of the more well known, and most quickly expanding, of these services. The WWW supported by the Internet is capable of combining technologies such as voice, video and text, and presenting them in an on-demand, interactive fashion to anyone connected to the Internet. In essence, the WWW allows anyone connected to the Internet with a computer that has the appropriate software and hardware configuration to retrieve any HyperMedia document that has been made available anywhere on the Internet. The HyperMedia document is received as a file representing a block of information, and within that file are hyperlinks to other documents (or blocks of information) which may, for example, contain texts, sounds, images or movies.

As the computer receiving the HyperMedia document constructs the document for display to the user, it may use some of these hyperlinks in order to receive other files necessary to complete the display. For instance, hyperlinks may be used to access image files which are to be

displayed as part of the document. Once the document has been constructed by the client computer and displayed to the user, the user may have the opportunity to select hyperlinks, for example by using a mouse or other input device to select a word or image on the screen, in order to request the document referenced by that Hyperlink.

As more and more people have started to use Internet services such as WWW, so the demands on the server computers has increased. In order to cope with these demands, parallel computers have started to be used as servers, these computers having a plurality of processors arranged to provide the service. By use of the multiple processors, very high volumes of server requests can be handled. An example of such a parallel computer is the IBM RISC System/6000 SP machines (IBM and RISC System/6000 are registered Trade Marks of International Business Machines Corporation).

However, a problem that arises in use of such parallel servers is that of balancing the incoming client traffic across the processors that are collaborating in the parallel system to serve documents (such as Web pages in WWW) and run service-based applications. With existing Load-Balancing mechanisms, there can be times when one particular processor becomes the focus of a large amount of traffic for a period of time. The technical term for this is that the processor gets "toasted". In this condition, the processor is overwhelmed by client requests, and grinds to a virtual halt, taking up to several minutes to service one request.

It is hence an object of the present invention to provide a mechanism to alleviate unbalanced load across the processors, thereby helping to avert this toasting effect.

Summary of the Invention

Accordingly the present invention provides a server computer connectable to a network and having a plurality of processors arranged to provide a service to one or more client computers connected to the network, the service comprising the provision of blocks of information to a client computer, and the processors having access to a data storage in which data for the blocks of information is stored, the server computer comprising: block retrieval means provided on each of said plurality of processors, and employed in response to a request from a client computer for a particular block of information from a processor, to retrieve data from the data storage in order to construct a file representing the requested block of information, the file including references to subsequent blocks of information requestable by the client; load determining means for periodically determining activity data for the

processors for inclusion in a load distribution record maintained for all of the processors of the server; decision means which, for each reference to a subsequent block of information in the file constructed by the block retrieval means, is arranged to determine, based on the load distribution record, which processor should service a request from the client computer for that subsequent block of information, and include an address for that processor in the file constructed by the block retrieval means; and delivery means for sending the file to the client computer.

By using the approach of the present invention, the contents of the file are dynamically altered before being sent to the client to include the addresses of processors from which any subsequent blocks of information should be retrieved. The decision as to which processor is to service a request for a subsequent block of information is taken based on the load distribution record for the processors of the server. In this way, a more effective balancing of the load across the processors is achievable, and hence problems such as "toasting" of a particular processor can be alleviated.

In preferred embodiments, the load determining means is provided on each of said plurality of processors to determine activity data for each processor. Further, the decision means is also preferably provided on each of said plurality of processors. However, as an alternative to the preferred embodiment, one or more of the load determining means, decision means and delivery means can be provided on a specific one of the plurality of processors, or on a completely independent processor dedicated to performing the function of those elements. Hence, a processor could be dedicated to performing the function of the decision means, and every file constructed by a block retrieval means could be sent to that processor for processing by the decision means.

In preferred embodiments, if a plurality of the subsequent blocks of information will be required by the client computer when constructing the block of information represented by the file, then the decision means is arranged to determine how many of those subsequent blocks of information any particular processor will service requests for, such that the greater the load on a processor as indicated by the load distribution record, the less the number of subsequent blocks of information that processor will service requests for. Typically, one or more of the subsequent blocks of information will be image files which will be required by the client computer when constructing the block of information.

In preferred embodiments, the network is the Internet, the service is the World Wide Web service, and the blocks of information are Web pages. However, it will be apparent that the invention is applicable to

any network and service wherein blocks of information are provided to a client which include references to subsequent blocks of information that the client may require, and which will need to be serviced by the server.

If the service is the World Wide Web service, then the address for a subsequent block of information included in the file by the decision means preferably takes the form of an address string including both an identifier for that block of information, and a unique identifier for the processor which the decision means has determined will service the request for that block of information.

Viewed from a second aspect, the present invention provides a method of operating a server computer connectable to a network and having a plurality of processors arranged to provide a service to one or more client computers connected to the network, the service comprising the provision of blocks of information to a client computer, and the processors having access to a data storage in which data for the blocks of information is stored, the method comprising the steps of: (a) in response to a request from a client computer for a particular block of information from a processor, employing that processor to retrieve data from the data storage in order to construct a file representing the requested block of information, the file including references to subsequent blocks of information requestable by the client; (b) employing a load determining means to periodically determine activity data for the processors for inclusion in a load distribution record maintained for all of the processors of the server; (c) for each reference to a subsequent block of information in the file constructed at step (a), determining, based on the load distribution record, which processor should service a request from the client computer for that subsequent block of information, and including an address for that processor in the file constructed at step (a); and (d) sending the file to the client computer.

Brief Description of the Drawings

The present invention will be described further, by way of example only, with reference to a preferred embodiment thereof as illustrated in the accompanying drawings, in which:

Figure 1 is a schematic block diagram illustrating a server in accordance with a preferred embodiment of the present invention;

Figure 2 is a flow diagram illustrating the processes carried out in the preferred embodiment by a processor of the server upon receipt of a request from a client; and

Figure 3 illustrates how the load distribution record is used to determine which processors will service follow-on requests for blocks of information.

Description of a Preferred Embodiment

In the preferred embodiment of the present invention, we will consider the WWW service provided over the Internet, and describe a mechanism for providing secondary load-balancing across a number of processors collaborating in a parallel Web server.

Reviewing first the structure of the WWW, this is based on Universal Resource Locators (URLs), which can be regarded as specifying a particular page or file on the WWW. The URL denotes both the server machine, and the particular file or page on that machine. For example, a link might be to "http://www.ibm.com/example.html". The "http:" indicates that we are using the World-Wide Web, the "www.ibm.com" is the Internet name of the Web server on the network that we want to request the page from, and the "example.html" is the name of the page we wish to receive.

The client computer employs software generally referred to as a browser to provide the user with a simple GUI, allowing the user to navigate over the WWW to different URLs. Sometimes a user will specify a particular URL by name, but more often a user will jump from one URL to another by means of references within a page (a so-called "hyperlink"). Thus a particular word or symbol on a page can be associated with another URL, so that selection of that word or symbol, eg by clicking a mouse at the relevant location, causes the client computer to retrieve and display the page from the selected URL. This new page may be located on a completely different server from the first page. A word or symbol which maps to another URL is sometimes referred to as a "hot button" (or similar). There may be many hot buttons or hyperlinks in a single page.

There may be many pages or URLs resident on a single server. If a user at a client computer selects a URL on a particular server computer, then the server computer performs an action specific to that page. In some cases this can lead to the server computer launching an application, but in the majority of cases the server computer retrieves the requested page from a database of stored pages, and this is transmitted back over the Internet to the client computer for display to the user.

The pages are formatted in accordance with a standard format known as HTML. A typical Web page contains some text content, some embedded images (graphics), and a number of links to other pages which the person viewing the page can click on if they want more information about a certain topic. The images are themselves Web "pages" which the Web browser automatically requests from the server computer on behalf of the client when it is downloading the page from the server computer. These images are typically stored as separate graphics files at the server, for example in standard ".gif" or ".jpeg" format. The HTML text contains references to these graphics files, which prompt the client to also retrieve the specified files from the server, thereby allowing the client to display both the HTML text and associated images. The follow-on links from the page can either be to some other page on the same server as the base page came from, or to any other Web server on the Internet. For the purposes of illustrating the preferred embodiment of the present invention, we are interested in the graphic files, and the follow-on links which are on the same server as the base page. We will refer to both of these items as referenced Web pages hereafter.

There are a variety of web browser software commercially available for use on client computers, for example Web Explorer from IBM Corporation, and Mosaic from NCSA. All such browsers understand HTML and other WWW standard formats and can display or output correctly files in these formats. Similarly, the mechanisms for navigating between different URLs are also understood and implemented by such programs. Further information about the World Wide Web can be found in "Spinning the Web" by Andrew Ford (International Thomson Publishing, London 1995) and "The World Wide Web Unleashed" by John December and Neil Randall (SAMS Publishing, Indianapolis 1994).

In preferred embodiment of the present invention, a load balancing technique is used in association with the parallel server to make a number of the processors of the server appear to share the same Internet name. For example, with reference to Figure 1, a parallel Web server 10, www.ibm.com, could be served by processors 20, 30, 40 called, say, www1.ibm.com, www2.ibm.com, and www3.ibm.com. The load-balancing software arranges for a request to www.ibm.com to be sent to one of the processors

which forms part of that parallel Web server. A suitable load-balancing technique is discussed in Published European Patent application EP-A-0,648,038, which describes a dynamic load balancing technique to enable the load on a parallel server to be balanced across the various processors (or computers) forming the server. When a program on a client computer 50 wishes to connect to a processor of the parallel server 10, it communicates with a data processing system 60, often called a name server, to obtain the network address for the desired server. According to the technique described in EP-A-0,648,038, decision logic 70 is provided to periodically study the processors or computers in the parallel server 10 and, based on some configurable criteria, to select one of those processors. The configurable criteria can be chosen such that the least heavily loaded processor at the time the criteria are applied will be selected by the decision logic 70. The address for this processor is then associated with a generic server name in storage 80 available to the name server 60, so that each time a client program requests a machine address using the generic server name, it is given the address of the processor in the parallel server 10 that was most recently chosen by the decision logic 70. Hence, if www.ibm.com is the generic name, a request from the client 50 to the name server 60 for the Internet address of www.ibm.com will result in the client being provided with the Internet address of one of the three processors 20, 30, 40. For more details of this technique, reference should be made to EP-A-0,648,038.

Traditionally, a follow-on link which referred back to the same server (often called a 'Web site') as the base page would reference the same Internet name as that of the base page. For example, on the page http://www.ibm.com/first_page.html, a follow-on link might be to http://www.ibm.com/second_page.html. Since the client computer has already queried the name server 60 for the Internet address of the generic name, www.ibm.com, it would typically just use the same internet address again. Hence the follow-on request will be directed at the same processor as serviced the original request. Since the name server 60 is not consulted again to resolve the generic name www.ibm.com, any load balancing techniques employed at the name server will not be utilised. As a result, the actual processor that serviced the original request may be inundated with requests for follow-on links, this resulting in the 'toasting' effect mentioned earlier.

It is possible that the client computer can be arranged so that it will always consult the name server 60 for follow-on requests. However, in an attempt to increase efficiency of name servers, name servers tend to cache details of recent name requests. Typically, the client will not communicate directly with the name server 60. Instead, it will generally communicate with its local name server, which will communicate with a series of other name servers, until finally the request is passed to the name server 60, this being the name server that has "authority" over the parallel Web server 10. The Internet address resolved by the name server 60 will then be passed back through all these name servers, each of which may choose to cache the specific internet address that www.ibm.com has been resolved to. Hence, if the client is arranged so that each time it needs to communicate with www.ibm.com, it does contact a name server, it will still tend to receive the address previously resolved by the name server 60 since, for example, the local name server might decide that it already knows the Internet address for that name and thus there is no need to pass the request on further. Hence, it can be seen that the load balancing technique on the name server 60, whilst useful for the first request by a client for the address of the server, is not useful for any subsequent follow-on links to the same generic server name.

In order to overcome this problem, the parallel Web server of the preferred embodiment is provided with the ability to modify the content of Web pages in the server as they are served, and takes advantage of that facility to modify the name of the processors to which the follow-on links refer. Considering the Figure 1 example, since www1.ibm.com, www2.ibm.com and www3.ibm.com are all servers in the www.ibm.com parallel Web server, a follow-on link which would otherwise refer to www.ibm.com can be changed to refer to a specific processor, for example www1.ibm.com. This forces the request for the follow-on page to go to a specific processor in the parallel Web server, and, as will be apparent, can be applied for all referenced Web pages, whether they be graphical images to be embedded in the currently displayed Web page, or follow-on links to different Web pages which the user can access from the displayed Web page.

By using the technique of the preferred embodiment to specify explicit processor names (eg. www1.ibm.com) in place of the generic name

for the whole parallel Server (www.ibm.com), some control can be exercised over the load which will hit the various processors of the server in the near future. This control enables the likelihood of 'toasting' to be reduced.

The system of the preferred embodiment will now be described in more detail with reference to Figure 1. In Figure 1, the functional elements of processor 30 required to implement the preferred embodiment are shown. Preferably, all the other processors 20, 40 of the server 10 that are to provide the WWW service include the same elements, but these have been omitted from the figure for the sake of simplicity. As an alternative to the preferred embodiment, one or more of the functional elements other than the block retrieval means can be provided on a specific one of the plurality of processors, or on a completely independent processor dedicated to performing the function of those elements. Hence, a processor could be dedicated to performing the function of the decision means, and every Web page retrieved by a block retrieval means could be sent to that processor for processing by the decision means.

If, after consulting the name server 60, the client computer 50 is provided with the Internet address of the processor 30, it will contact the processor 30 directly with its request for a Web page. The block retrieval means 80 will access data in the storage device 120 in order to construct a file representing the Web page requested by the client 50. For any Web page referenced in that file, the decision means 90 will be employed to determine which specific processor should be chosen to handle subsequent requests for that Web page, and an identifier for that processor (eg www1.ibm.com) will be included at the appropriate point in the file.

As will be discussed in more detail later, when determining which processor will handle a follow-on request, the decision means 90 refers to a load distribution record providing information about the loads on all the processors of the server. The load determining means 100 is arranged to periodically assess the activity of the processor 30, and provide that information for inclusion in the load distribution record.

Once the decision means has determined which processors will handle any referenced web pages, and that information has been included in the file constructed by the block retrieval means, the file is passed to the delivery means 110 for sending to the browser 130 of the client computer 50. The browser 130 will then construct the Web page from the file for presentation to the user of the client computer 50, and will send requests for referenced Web pages directly to the processors identified in the file.

The manner in which the load distribution record is obtained will now be discussed. Statistics are gathered by the parallel Web server about how busy the processors of that server are, and these can then be collated to produce a "load distribution record". It will be apparent that this can be done in a number of ways. For instance, each processor can include a routine which periodically determines how busy that processor is, and then sends that information to some central location for inclusion in the load distribution record. The load distribution record may be kept on the data storage device 120 for access by each processor, or could instead be kept on the server in a portion of shared memory accessible by all the processors, if the architecture of the server supports such a shared memory arrangement. Alternatively, a series of messages can be sent over a communications network to each processor, such that each processor periodically assesses how busy it is, and then broadcasts that information to all the other processors in the parallel Web server. Each processor can then keep the load distribution record in its own local memory, and update it whenever it receives new load data from the other processors.

The decision means 90 of each processor 20, 30, 40 then uses this load distribution record to dynamically modify the Internet names of the follow-on links and images referenced in the Web pages that it subsequently serves. The profile is updated periodically, at a frequency which is configurable by the Web server administrator. The effect of doing this modification of content is that a processor which is beginning to get busy will have proportionately fewer follow-on requests directed to it, thus averting the serious performance impact when a processor gets "toasted". As the processor recovers, it will have more requests directed to it, and so the monitoring and load-tuning cycle will continue.

The process carried out by a processor in accordance with the preferred embodiment upon receipt of a request from a client will now be described with reference to Figure 2. At step 200, the processor awaits a request from a client. At step 210, a request 205 from a browser of a client computer is received. The processor then determines which Web page has been requested at step 220, and the block retrieval means 80 then fetches that page from the database 120 and constructs a file (step 230). Next, at step 240, any references to other web pages in that file are identified and the decision means 90 uses the latest load distribution record 270 to determine for each of those Web pages which processor should service a request for that web page. Hence, if a follow on Web page is called 'second_page.html', and the decision means determines that the processor 20 should service the request, the reference in the file will be changed to read 'http://www1.ibm.com/second_page.html'.

Once these follow-on hyperlinks have been completed by the decision means, the file is then sent to the client browser at step 250. Next, in preferred embodiments, the processor is arranged to reassess how busy it is at step 260, updating the load distribution record based on that assessment. Once this has been done, the processor returns to step 200 where it awaits the next request from a client.

As mentioned earlier, the follow-on Web pages referenced in the file may be categorised in to two types. Some may refer to Web pages which the browser will need to retrieve during the construction of the current web page for display to the user. For example, the basic file might include all the text for the Web page, but the data for any images to be displayed will not typically be included, but rather, references to Web pages including the images will be provided in the file returned to the browser. The browser will need to retrieve these web pages in order to reproduce the complete web page for the user.

The second type of Web page referenced in the file will be follow-on links which are not required by the browser in order to produce the initial web page for display. Instead, these will be pages which the user can select once the Web page has been displayed. Typically, the user will be able to select particular highlighted words or images on the screen, these words and images being associated with other Web pages. Upon such a

selection, the browser will retrieve from the file the address of the processor which is to service requests for that Web page and will contact that processor directly.

With particular regard to the first type of follow-on Web page identified above, Figure 3 illustrates how the load distribution record can be used by the decision means 90 to split requests for this first type of Web page amongst the available processors of the server. Since this first type of web page will be required almost immediately by the browser so that it can construct the complete web page for display to the user, then it can be predicted that these requests will come back to the server fairly shortly after the original web page is delivered to the browser.

In Figure 3, the load on each processor as a percentage of the maximum load is plotted for each processor. Typically the maximum load is determined as the point at which the processor is unable to provide an acceptable response time, this generally being determined by the server administrator. It will be apparent that there are clearly other ways in which the load can be assessed. From Figure 3, it can be seen that processor 2 is most heavily loaded, followed by processor 3, whilst processor 1 is the least heavily loaded. If, for example, there are nine follow-on Web pages of the first type, then the decision means can use the percentage load figures for each processor to determine how many of those nine web pages should be serviced by each processor.

As illustrated in Figure 3, the unused load for each processor has been divided into equal size blocks, such that a total of nine blocks are produced. A suitable way of balancing the load across the processors is to assign a particular processor the responsibility of servicing follow-on requests for a number of Web pages equal to the number of blocks associated with the unused load of that processor. Hence, in the example illustrated in Figure 3, the address of processor 1 will be referenced for five Web pages, the address of processor 2 will be referenced for one Web page, and the address of processor 3 will be referenced for three Web pages.

From the above description, it will be apparent that in accordance with the preferred embodiments of the invention, the content of Web pages is dynamically altered as they are served, to change the Internet name of the server processor from which any follow-on link should be retrieved. The processor is chosen from a set of processors which can access the same Web content, and the choice is made based on the relative loading of those servers. In this way, a more effective load balance is achieved, and problems of processor "toasting" are averted before they become serious.

CLAIMS

1. A server computer connectable to a network and having a plurality of processors (20, 30, 40) arranged to provide a service to one or more client computers connected to the network, the service comprising the provision of blocks of information to a client computer (50), and the processors having access to a data storage (120) in which data for the blocks of information is stored, the server computer comprising:

block retrieval means (80) provided on each of said plurality of processors, and employed in response to a request from a client computer (50) for a particular block of information from a processor, to retrieve data from the data storage (120) in order to construct a file representing the requested block of information, the file including references to subsequent blocks of information requestable by the client (50);

load determining means (100) for periodically determining activity data for the processors for inclusion in a load distribution record maintained for all of the processors of the server;

decision means (90) which, for each reference to a subsequent block of information in the file constructed by the block retrieval means (80), is arranged to determine, based on the load distribution record, which processor should service a request from the client computer (50) for that subsequent block of information, and include an address for that processor in the file constructed by the block retrieval means (80); and

delivery means (110) for sending the file to the client computer (50).

2. A server computer as claimed in Claim 1, wherein the load determining means (100) is provided on each of said plurality of processors to determine activity data for each processor.

3. A server computer as claimed in Claim 1 or Claim 2, wherein the decision means (90) is provided on each of said plurality of processors.

4. A server computer as claimed in any preceding claim, wherein if a plurality of the subsequent blocks of information will be required by the client computer (50) when constructing the block of information represented by the file, then the decision means (90) is arranged to determine how many of those subsequent blocks of information any particular processor will service requests for, such that the greater the load on a processor as indicated by the load distribution record, the less the number of subsequent blocks of information that processor will service requests for.

5. A server as claimed in any preceding claim, wherein one or more of the subsequent blocks of information are image files which will be required by the client computer (50) when constructing the block of information.

6. A server as claimed in any preceding claim, wherein the network is the Internet.

7. A server as claimed in Claim 6, wherein the service is the World Wide Web service, and the blocks of information are Web pages.

8. A server as claimed in claim 7, wherein the address for a subsequent block of information included in the file by the decision means (90) takes the form of an address string including both an identifier for that block of information, and a unique identifier for the processor which the decision means (90) has determined will service the request for that block of information.

9. A method of operating a server computer connectable to a network and having a plurality of processors (20, 30, 40) arranged to provide a service to one or more client computers connected to the network, the service comprising the provision of blocks of information to a client computer (50), and the processors having access to a data storage (120) in which data for the blocks of information is stored, the method comprising the steps of:

(a) in response to a request from a client computer (50) for a particular block of information from a processor, employing that

processor to retrieve data from the data storage (120) in order to construct a file representing the requested block of information, the file including references to subsequent blocks of information requestable by the client (50);

(b) employing a load determining means (100) to periodically determine activity data for the processors for inclusion in a load distribution record maintained for all of the processors of the server;

(c) for each reference to a subsequent block of information in the file constructed at step (a), determining, based on the load distribution record, which processor should service a request from the client computer (50) for that subsequent block of information, and including an address for that processor in the file constructed at step (a); and

(d) sending the file to the client computer (50).

10. A method as claimed in Claim 9, wherein each of said plurality of processors includes a load determining means (100) arranged to perform said step (b).

11. A method as claimed in Claim 9 or Claim 10, wherein each of said processors includes a decision means (90) arranged to perform said step (c).

12. A method as claimed in and of claims 9 to 11, wherein if a plurality of the subsequent blocks of information will be required by the client computer (50) when constructing the block of information represented by the file, then said step (c) includes determining how many of those subsequent blocks of information any particular processor will service requests for, such that the greater the load on a processor as indicated by the load distribution record, the less the number of subsequent blocks of information that processor will service requests for.

13. A method as claimed in any of claims 9 to 12, wherein the address for a subsequent block of information included in the file at said step (c) takes the form of an address string including both an identifier for

that block of information, and a unique identifier for the processor which at step (c) has been determined as the processor which will service the request for that block of information.

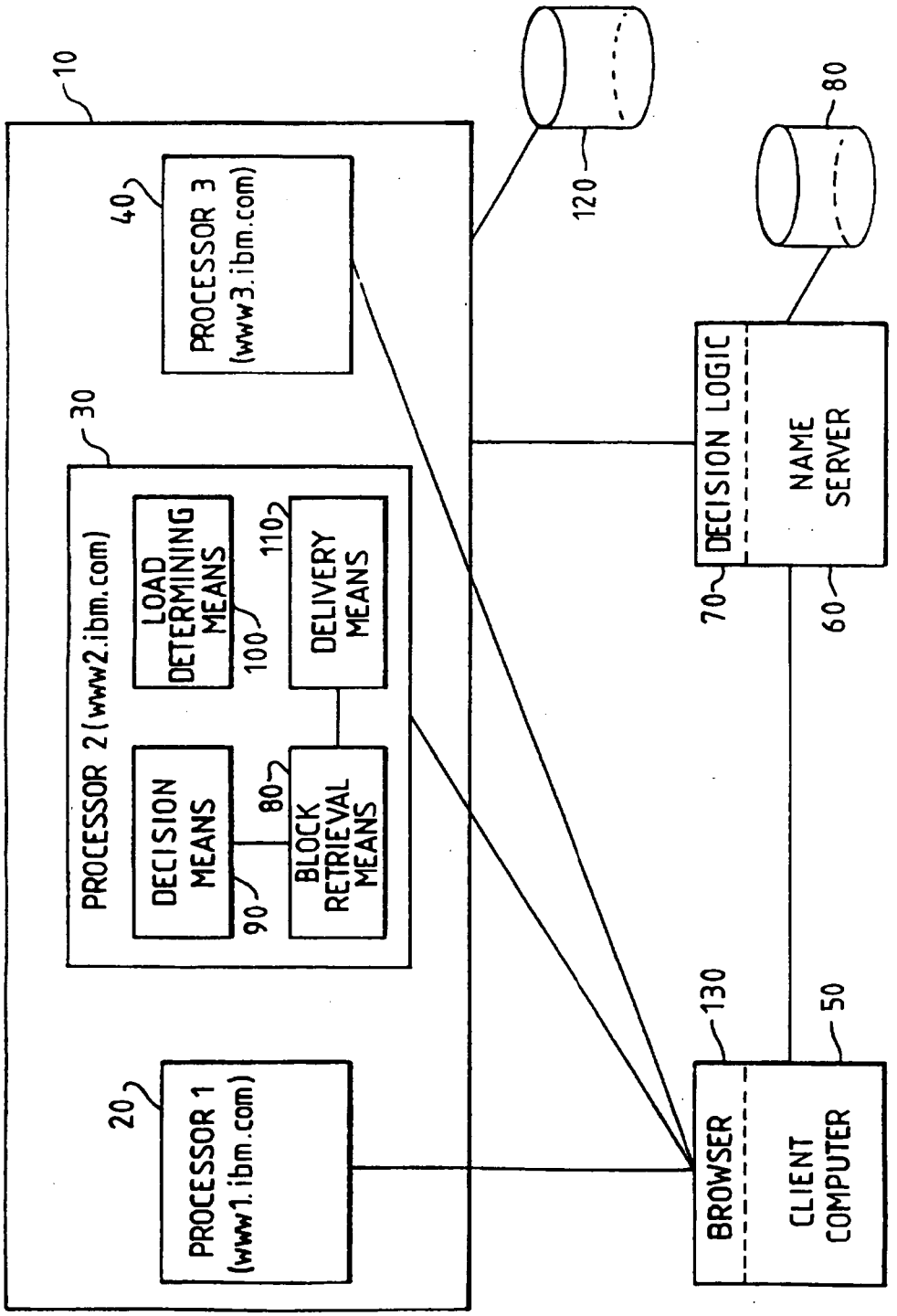

10
PROCESSOR 3 (www3.ibm.com)
40
120
80
30
PROCESSOR 2 (www2.ibm.com)
LOAD DETERMINING MEANS
100
DELIVERY MEANS
110
DECISION MEANS
90
BLOCK RETRIEVAL MEANS
80
DECISION LOGIC
NAME SERVER
70
60
PROCESSOR 1 (www1.ibm.com)
20
BROWSER
130
CLIENT COMPUTER
50

FIG. 1

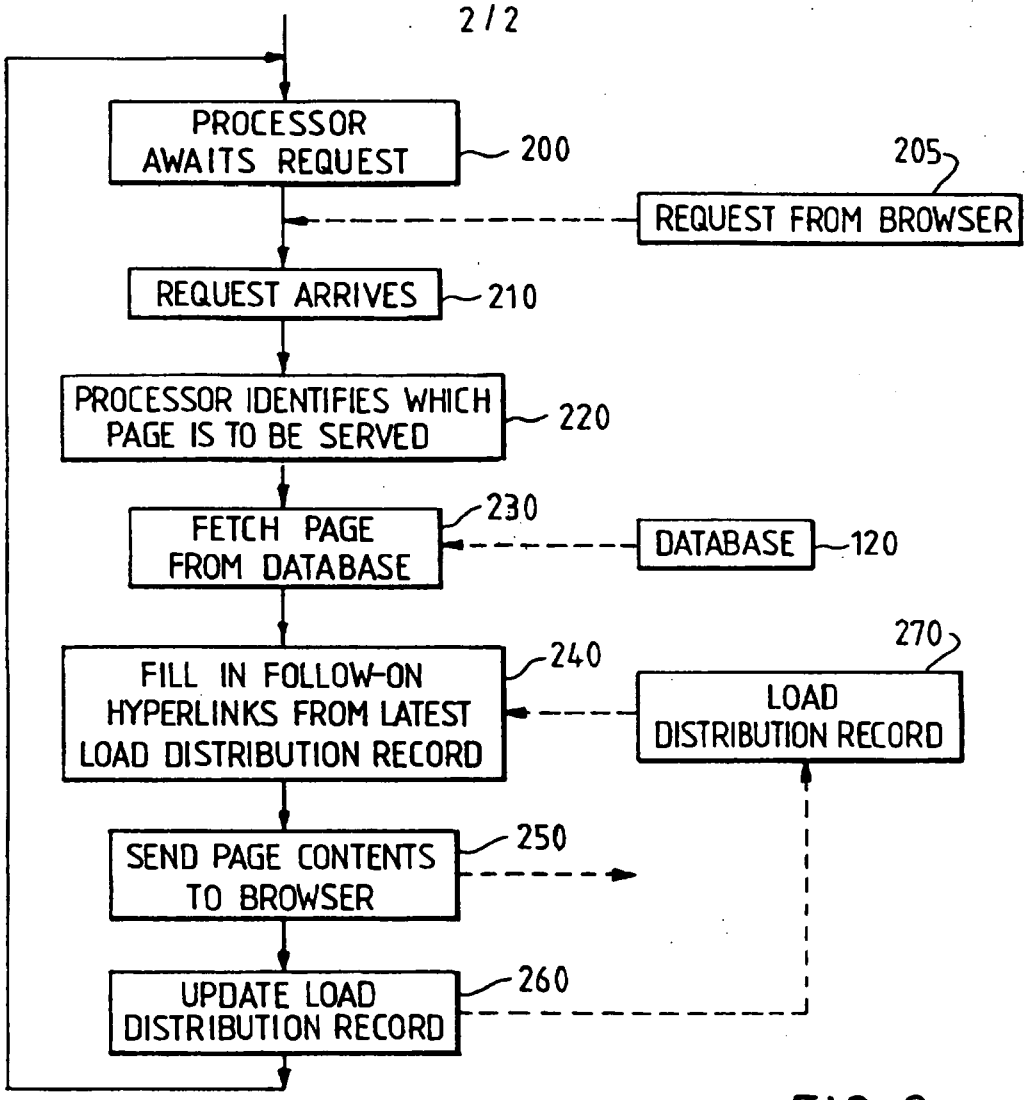

PROCESSOR AWAITS REQUEST
200
205
REQUEST FROM BROWSER
REQUEST ARRIVES
210
PROCESSOR IDENTIFIES WHICH PAGE IS TO BE SERVED
220
FETCH PAGE FROM DATABASE
230
DATABASE
120
FILL IN FOLLOW-ON HYPERLINKS FROM LATEST LOAD DISTRIBUTION RECORD
240
270
LOAD DISTRIBUTION RECORD
SEND PAGE CONTENTS TO BROWSER
250
UPDATE LOAD DISTRIBUTION RECORD
260

FIG. 2

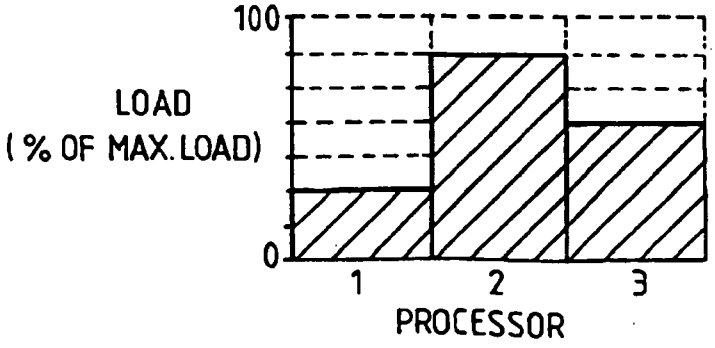

100
LOAD
(% OF MAX. LOAD)
0
1
2
3
PROCESSOR

FIG. 3

INTERNATIONAL SEARCH REPORT

Inte onal Application No
PCT/GB 96/02983

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/46 G06F17/30 H04L29/06

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	PROCEEDINGS, WORKSHOP ON MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 8 December 1994, pages 185-190, XP000602490 VOELKER G M ET AL: "MOBISAIC: AN INFORMATION SYSTEM FOR A MOBILE WIRELESS COMPUTING ENVIRONMENT" see page 187, paragraph 3.1 ---	1,5-9
A	EP 0 648 038 A (IBM) 12 April 1995 cited in the application see the whole document --- -/--	1-4,9-13

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

1

Date of the actual completion of the international search	Date of mailing of the international search report
12 March 1997	27.03.97

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax (+31-70) 340-3016	Michel, T

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inte onal Application No
PCT/GB 96/02983

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DATABASE, vol. 18, no. 1, 1 February 1995, pages 54-59, XP000571798 POWELL J: "SPINNING THE WORLD-WIDE WEB AN HTML PRIMER" see page 57, left-hand column, line 43 - right-hand column, line 35 ---	1,5-9
A	COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 1/02, 1 December 1995, pages 155-160, XP000542209 ROBERTON D ET AL: "VIRTUAL FROG DISSECTION: INTERACTIVE 3D GRAPHICS VIA THE WEB" see page 158, left-hand column, line 20 - right-hand column, line 4 -----	1,5-9

1

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inte onal Application No
PCT/GB 96/02983

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0648038 A	12-04-95	GB 2281793 A	15-03-95
		JP 2505116 B	05-06-96
		JP 7093238 A	07-04-95

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/00 — A1

(11) International Publication Number: WO 97/42582

(43) International Publication Date: 13 November 1997 (13.11.97)

(21) International Application Number: PCT/US97/07893

(22) International Filing Date: 8 May 1997 (08.05.97)

(30) Priority Data:
08/644,072 9 May 1996 (09.05.96) US

(71) Applicant: NETCAST COMMUNICATIONS CORP. [US/US]; 9 DesBrosses Street, New York, NY 10013 (US).

(72) Inventors: MONTEIRO, Antonio, M.; Apartment 631, 21 South End Avenue, New York, NY 10280 (US). BUTTERWORTH, James, F.; 32 N. Moore Street, New York, NY 10013 (US).

(74) Agents: MORRIS, Francis, E. et al.; Pennie & Edmonds LLP, 1155 Avenue of the Americas, New York, NY 10036 (US).

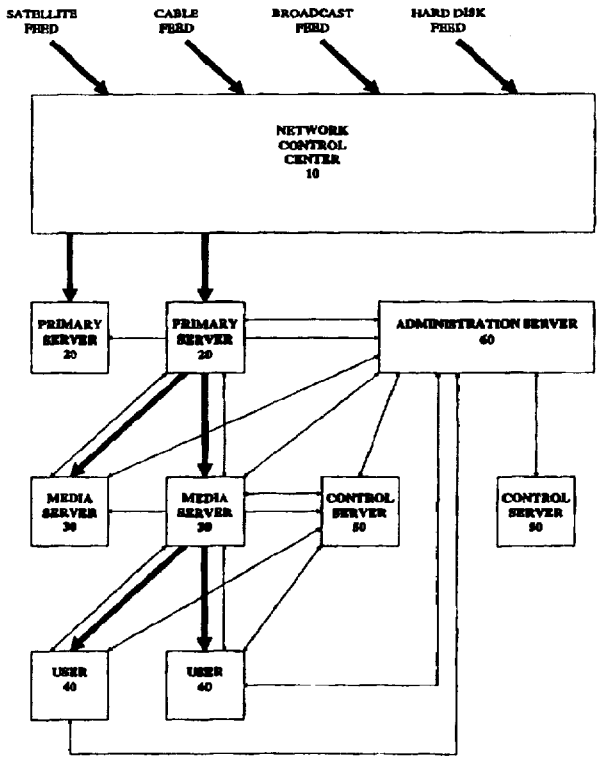
(81) Designated States: AU, BR, CA, JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: MULTICASTING METHOD AND APPARATUS

(57) Abstract

A scalable architecture is disclosed for delivery of real-time information over a communications network. Embedded into the architecture is a control mechanism (10) that provides for the management and administration (60) of users (40) who are to receive the real-time information. Preferably, there are multiple channels of information available simultaneously to be delivered to users, each channel consisting of an independent stream of information. A user (40) chooses to tune in or tune out a particular channel, but does not choose the time at which the channel distributes its information. Advantageously, interactive (two-way) information can be incorporated into the system, multiple streams of information can be integrated for delivery to a user (40), and certain portions of the information being delivered can be tailored to the individual user (40).



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

MULTICASTING METHOD AND APPARATUS

1. Field of the Invention

This relates to a method and apparatus for providing audio and/or visual communication services, in real-time to a multiplicity of identifiable users on a communications network, such as the Internet. In a preferred embodiment, the invention monitors which users are receiving signals on which one of a plurality of channels and modifies the content of at least some signals in response thereto. A particular application is to provide services akin to multi-channel radio or television with commercial programming content adjusted in accordance with the identity of the individual user.

2. Background of the Invention

Systems such as the Internet typically are point-to-point (or unicast) systems in which a message is converted into a series of addressed packets which are routed from a source node through a plurality of routers to a destination node. In most communication protocols the packet includes a header which contains the addresses of the source and the destination nodes as well as a sequence number which specifies the packet's order in the message.

In general, these systems do not have the capability of broadcasting a message from a source node to all the other nodes in the network because such a capability is rarely of much use and could easily overload the network. However, there are situations where it is desirable for one node to communicate with some subset of all the nodes. For example, multi-party conferencing capability analogous to that found in the public telephone system and broadcasting to a limited number of nodes are of considerable interest to users of packet-switched networks. To satisfy such demands, packets destined for several recipients have been encapsulated in a unicast packet and forwarded from a source to a point in a network where the packets have been

replicated and forwarded on to all desired recipients. This technique is known as IP Multicasting and the network over which such packets are routed is referred to as the Multicast Backbone or MBONE. More recently, routers have become available which can route the multicast addresses (class D addresses) provided for in communication protocols such as TCP/IP and UDP/IP. A multicast address is essentially an address for a group of host computers who have indicated their desire to participate in that group. Thus, a multicast packet can be routed from a source node through a plurality of multicast routers (or mrouters) to one or more devices receiving the multicast packets. From there the packet is distributed to all the host computers that are members of the multicast group.

These techniques have been used to provide on the Internet audio and video conferencing as well as radio-like broadcasting to groups of interested parties. See, for example, K. Savetz *et al.* MBONE Multicasting Tomorrow's Internet (IDG Books WorldWide Inc., 1996).

Further details concerning technical aspects of multicasting may be found in the Internet documents Request for Comments (RFC) 1112 and 1458 which are reproduced at Appendices A and B of the Savetz book and in D.P. Brutaman *et al.*, "MBONE provides Audio and Video Across the Internet," IEEE Computer, Vol. 27, No. 4, pp. 30-36 (April 1994), all of which are incorporated herein by reference.

Citation of the foregoing documents is not to be construed as an admission that any of such documents is a prior art publication relative to the present invention.

3. Summary of the Invention

The present invention is a scalable architecture for delivery of real-time information over a communications network. Embedded into the architecture is a control mechanism that provides for the management and administration of users who are to receive the real-time information.

In the preferred embodiment, the information being delivered is high-quality audio. However, it could also be video, graphics, text or any other type of information that can be transmitted over a digital network. This information is delivered in real-time to any number of widely distributed users. It is *real-time* in that for a given channel of information, approximately the same information is being sent at approximately the same time to everyone who is enabled to receive the information.

Preferably, there are multiple *channels* of information available simultaneously to be delivered to users, each channel consisting of an independent stream of information. A user chooses to tune in or tune out a particular channel, but does not choose the time at which the channel distributes its information. Advantageously, interactive (two-way) information can be incorporated into the system, multiple streams of information can be integrated for delivery to a user, and certain portions of the information being delivered can be tailored to the individual user.

4. Brief Description of the Drawing

These and other objects, features and advantages of our invention will be more readily apparent from the following Detailed Description of a Preferred Embodiment of our invention in which

Fig. 1 is a schematic diagram depicting an overview of the system of the present invention;

Fig. 2 is a schematic diagram depicting the network control center for the system of Fig. 1;

Fig. 3 is a schematic diagram depicting a unicast distribution structure;

Fig. 4 is a schematic diagram depicting a multicast distribution structure;

Fig. 5 is a schematic diagram depicting the connection between the media server and the user in the system of Fig. 1;

Figs. 6-17 are timing diagrams which depict various aspects of the operation of the system of Fig. 1; and

Figs. 18 and 19 depict the user interface for control of the system of Fig. 1.

Where the same reference numerals appear in multiple drawings, the numerals refer to the same or corresponding structure in such drawings.

5. Detailed Description of the Preferred Embodiment

Referring to Fig. 1, the system of the present invention comprises a Network Control Center 10, a plurality of Primary Servers 20, Media Servers 30, Users 40 and Control Servers 50 and an Administration Server 60. The servers are interconnected by a communications network, which in the preferred embodiment is the global connected internetwork known as the *Internet*. The Network Control Center 10 is the source of the information being distributed. It receives audio feeds from satellite, over the air broadcast or in other ways and processes this information for delivery over the network on multiple channels of information. This processing consists of optionally recording the information for future broadcast and dynamically inserting paid commercial advertisements.

For each channel of information, there is a Primary Server 20 that receives the stream of information from the Network Control Center 10 and compresses the information stream to allow for more efficient transmission. The Primary Servers 20 are directly connected to the network.

The Primary Servers forward information via the network to a number of Media Servers 30. There may be a large number of Media Servers and in fact there may be many levels of Media Servers. For example, a Media Server which receives a stream of information from a Primary Server may forward that stream via the network to another Media Server which then forwards it to a User 40. This multilevel hierarchical structure is described in more detail below.

The topology of the Internet dictates the ideal placement of Media Servers, the fan-out of each Media Server and the number of levels of Media Servers between the Primary Server and Users. For example, the Media Servers which feed from a Primary Server might be placed at a major points of presence (POPs) of each of the large Internet service providers. These Media Servers might also be placed near *clouds* which serve as high bandwidth exchange points between the major carriers. Similarly, Media Servers which feed to Users might be placed on or close to networks which have a large number of subscribers to minimize the distance and number of data streams being transmitted.

Control Servers 50 are responsible for keeping track of which Users are listening to which channels and for directing the Media Servers to start and stop streams of information to those Users. The Control Servers are also responsible for handling other interactions among the various components of the system as will be described in more detail below. Each Control Server is responsible for managing a *cluster* of Media Servers; and each Media Server is managed by a single Control Server at any given time. As a result, the Control Servers are distributed throughout the Internet, preferably located close to the Media Servers.

The Administration Server 60 is responsible for registering new Users, authenticating Users who want to log onto the system, and maintaining audit logs for how many Users are listening to which channels and at which times. Maintaining audit logs and gathering statistics are features critical to monitoring the delivery of paid commercial messages as well as for other purposes. For example, for purposes of assessing copyright royalties, the audit logs can record the number of listeners for each musical or video selection that is distributed by the system. Another application is to determine the percentage of listeners who are interested in listening to a particular musical selection by determining how many listen to the entire selection and how many turn it off.

The system of the present invention can be considered a distribution architecture integrated with a control architecture. The distribution architecture handles scalable real-time delivery of information to any number of Users on a packet switched network, such as the Internet. The control architecture represents a second scalable system integrated with the distribution architecture for managing and administering the delivery of that information.

The remainder of this description is divided into three sections. In the next section the distribution architecture will be described in more detail. Following that, the control architecture will be described. In the third section the User interface will be illustrated.

I. Distribution Architecture

The distribution architecture provides for the delivery of real-time information to any number of Users distributed throughout a network. As will be described in detail below, the distribution architecture is scalable to allow for efficient delivery of multiple simultaneous information channels in real-time to a large number of Users.

In the preferred embodiment, the information which is being distributed consists of high-quality audio in addition to other information. It should be appreciated that the basic architecture and other general principles set forth herein would also apply to the delivery of video, graphics, text or any other type of information that can be delivered over a digital network. In addition, it should be appreciated that an information stream can consist of audio with supplemental information such as text and graphic images and commands to control software running on the User's computer.

The source of information in the preferred embodiment is the Network Control Center 10, depicted in the schematic diagram of Fig. 2. Control Centers of this type of design are available from Broadcast Electronics, Inc. and are

similar to what would be found in a conventional radio station serving multiple frequencies.

Referring to Fig. 2, the incoming signal can be received in a variety of ways such as from a satellite, over-the-air broadcast, cable or hard disk. It is then processed by Receiver/Decoder 110, which decodes the signal and provides an incoming audio stream. Routing Switcher 120 is responsible for routing the incoming audio feed from the Receiver to either Delay Recording Workstation 140 or to one of the Playback/Control Workstations 130. Real-time insertion of paid commercial advertising takes place at the Playback/Control Workstations and the resulting integrated audio stream is delivered to the Primary Servers. The Delay Recording Workstation is responsible for recording an incoming broadcast so that it can be played back at a later time.

Supervisory Workstation 150 is responsible for managing and controlling the Playback/Control Workstations, Delay Recording Workstations and other computers as may be connected to the local area network within the Network Control Center. Production Workstation 160 and AudioVAULT-NFS Server 170 are used to manipulate audio samples, such as commercial messages for use by the Playback/Control Workstations. The audio being delivered can consist of syndicated TV or radio programs, such as would be received over satellite or cable and delivered as described above. These can be delivered live and/or played back at a later time. It is also possible for the delivery of information, such as music, to take place from information that is all stored locally such as on a hard disk. A new play list and its associated music data can then be downloaded periodically to update the channel. Additionally, it is possible to deliver commercial-free programming, for example public service announcements or label-specific music.

In the preferred embodiment the Primary Servers are responsible for compressing the audio stream using an advanced perceptual technique developed and licensed by AT&T

Corp. and Lucent Technologies, Inc. This highly sophisticated algorithm is used to maximize the benefit of the bandwidth available. Advantageously, two bitrates are available, a first rate of approximately 20Kbps and a second rate of approximately 56Kbps. Using the perceptual technique, the quality of the first rate is similar to FM monaural (with a sampling rate of approximately 22,000 16-bit samples per second) and the second rate is close to CD quality stereo (with a sampling rate of approximately 32,000 16-bit samples in stereo each second). The signals at the two different bitrates comprise two different audio channels and thus require two different compression processes.

The computational requirements of compressing an audio stream in real time using techniques such as the advanced perceptual technique are approximately 100% of a Pentium-Pro 200Mhz computer and the computational requirements of decompressing an audio stream in real time are approximately 30% of a Pentium 75Mhz computer. Future improvements and/or changes to the algorithm could significantly change these requirements. For the present, a dedicated computer is required within the Primary Server to compress the audio stream. The decompression process takes place on end Users' computers and preferably would use only a portion of the computers' computational requirements, allowing the computers to be used for other tasks while they are processing the audio stream.

It is important to appreciate that the compression and decompression techniques employed by the present invention are not critical to the overall operation of the system and the advantages obtained therefrom could be obtained with other compression methodologies. Advantageously, the identity of the compression technique used can be encoded into the audio stream in the packet header. This makes it possible to identify to the receiver the nature of the decompression algorithm to use; and thereby make it possible for the computer within the Primary Server

to select an optimum compression algorithm depending on the nature of the audio stream to be compressed.

The remainder of the distribution architecture comprises the multilevel hierarchy of data transmission originating at the Primary Server 20 and terminating at the Users 40 as shown in Figure 3. In the preferred embodiment, the network is the global connected Internet. It can also include private networks which are connected to the Internet and it could be implemented on any packet switched network, cable-modem-based or satellite-based cable system. It is possible that certain links within the overall system, for example, the link between the Primary Server and the first level of Media Servers, are private data links which carry only data associated with this system. This could also be true of other data transmission paths in the distribution architecture. The User receiving the information preferably can be anyone who has access to the Internet with sufficient bandwidth to receive the resulting audio data.

It should be appreciated that the distribution architecture of the present invention provides for scalability. Using such a structure, any number of Users, and as widely distributed as necessary, can be accommodated. In the preferred embodiment, the fan-out at each level of Media Server (given the state of technology today) is on the order of ten, but the same structure could be applied with other fan-outs. The location and fan-out of the Media Servers is chosen to minimize overall network bandwidth consumed.

The flow of information from Primary Server 20 through network to User 40 is based on the delivery of a continuous sequence of individual pieces of information, or packets. Thus the distribution architecture implements a form of multicast packet delivery to a group. The group in this case is the set of all Users who are listening to a given channel at a given time. Group membership is dynamic, Users can start and stop listening to a channel at any time.

Multicasting can be implemented in a variety of ways, any or all of which can be used in the present invention. In the preferred embodiment, the Media Servers receive unicast packet streams and they then duplicate these streams into more unicast streams to other Media Servers which are in the membership group for that stream. The lowest level Media Servers use hardware broadcast, multicast and/or unicast to reach all Users served by that Media Server.

If the Media Server is directly connected to the same physical network as the User, hardware broadcast or multicast can be used to transmit the packet stream to all Users listening at that time on that network. In this case the Media Servers can translate the incoming packets into broadcast or multicast packets for transmission on the local network. Only a single packet is transmitted at-a-time on the local network and any computer directly connected to the local network can receive that packet. Hardware multicast is built into most networks and it is lower in overall overhead than hardware broadcast since computers not interested in a transmission do not have to process the packets. In the case that a Media Server is serving a User who is not on the same physical network, a unicast transmission is used to reach that User, which requires a separate packet transmission for each User so connected. In the preferred embodiment, the assignment of Users to Media Servers is done using control transactions among the User 40, Control Servers 50, and Administration Server 60. This system will be described more fully in the following section.

Multicasting can also be implemented within the Internet at the IP level using IP class D addresses and the IGMP group control protocol. Fig. 4 illustrates how the multilevel hierarchical distribution architecture would operate using IP multicast delivery. Under this system, a packet is transmitted with a multicast address for a destination and each router maintains group membership lists for each interface that it is connected to and will forward

packets across the Internet to other routers such that all Users within the global group eventually receive a copy of the packet. Unless and until all routers within the Internet understand multicasting in this way, it is necessary to supplement it with IP tunneling in which multicast packets are encapsulated in unicast packets and routed by unicast routers to a multicast routers. The present invention can and will be able to take advantage of IP multicasting as it becomes widely available. Each channel of information would be given its own class D address and the Media Server would then simply transmit packets using the appropriate IP destination address. In this case no Media Servers would be used as this function would be accomplished by the routers in use to store and forward other IP packets.

Thus it can be appreciated that the implementation of the multicast delivery structure can be implemented using a combination of IP unicast, IP multicast and hardware multicast or any other system which provides for distributed delivery of information to a specific group of destinations. It is expected that special relationships with Internet providers will be established so that delivery of the audio steams can take place with a guaranteed bandwidth and in the most efficient way possible.

In the preferred embodiment, packets of information for distribution use the UDP protocol under IP rather than the TCP protocol. TCP provides for reliable stream delivery but at the cost of retransmission and delays. For real-time information, it is usually more appropriate to use UDP since the information is time critical and low latency is more important that reliability. Since TCP is a point-to-point protocol, it is incompatible with IP multicasting. However, TCP could be used on the IP unicast links between Media Servers which are expected to have very low packet loss. In order to handle out of order, lost, duplicate and corrupted packets, the UDP packets are serialized.

In the preferred embodiment the size of the audio packets being transmitted is variable and can change on a

packet by packet basis. It is expected that when using compression schemes that have a fixed bit rate, such as ADPCM, all packets for that stream would be the same size. Alternatively when using a variable bit rate compression algorithm, it is expected that packet size would vary so as to establish approximately the same amount of time for each sample. For example, if each packet corresponds to a 20 millisecond segment of speech, this could correspond to 100 bytes during one time period and 200 bytes during another. Additionally, the Media Server may choose to dynamically vary the packet size to accommodate changes in network conditions.

Since the resulting playback of audio information is sensitive to packet loss and network congestion, software running on the various computers which make up this system monitor the ongoing situation and adapt to it in the best possible way. This may involve using different Media Servers and/or lowering the data rate to the User. For example, similar to analog dynamic signal quality negotiation present in many analog radio receivers, the User software may request a lower bitrate until the situation is improved. Also, note that the audio information being delivered to the User is preferably interleaved so that a contiguous segment of the audiostream is distributed for transmission over several packets. As a result, the loss of one packet is spread out over multiple audio samples and causes minimal degradation in audio. Advantageously, a small degree of redundancy may be incorporated within the audio stream to further guard against packet loss.

Preferably, there are two bitrate options available to the User for audio delivery. These are approximately 20Kbps for standard audio and approximately 56Kbps for high quality audio. Thus, a 28.8Kbps modem connection over an analog phone line is sufficient to listen to standard audio broadcasts. To listen to high quality audio, an ISDN connection to the Internet is required, or some other connection with greater than 56Kbps bandwidth. It should be appreciated that higher bandwidths are currently becoming

available to end Users. In particular the use of cable modems and residential fiber networks are enhancing the bandwidths available to Users and thus making broadcasts of higher bitrates more practical.

In addition to the content of the audio channel being delivered, it is also possible to deliver out of band of *side-bar* information such as graphics, images and text. This side-bar information is synchronized with the audio channel. This may only involve small increases in bandwidth requirements, such as 1-2Kbps. For example a music program could deliver images of an album cover, the text of song lyrics, or URLs for use by a Web browser. The User can preferably choose to have the side-bar information show up automatically or be hidden. It is also possible to incorporate two-way interaction into the system, such that for example Users can participate in a global *chat* session during the audio broadcast. These and other details are explained in more detail below under the description of the User interface.

The delivery of paid commercial advertising information is an important aspect of the present invention. Advertising may be incorporated into the audio stream within the Network Control Center as described above. It may also be incorporated into the audio stream at the User level, or at some intermediate point in the distribution architecture. In addition, the side-bar information discussed above can also include advertising content. Fig. 5 illustrates the provision to the User of two separate streams 32, 34 of packets, one of which may be used for advertising. In this case the insertion of the stream of commercial advertising into the non-commercial stream occurs on the User's computer. Fig. 5 also illustrates packet stream 36 which identifies the User to the system. This enables the system to monitor which Users are listening to which channels and also allows the system to vary, for example, the advertising content delivered to a User.

One advantage of this alternative is to allow targeted commercial delivery based on the individual User. That is, an individual User would receive the main audio feed plus a particular advertising stream unique to his demographic group. Note that the advertising stream typically is lower in overall bitrate and generally does not require real-time delivery, thus lowering the overall load on the network. For example, the advertising stream could be delivered to the User in advance of the regular programming, stored in a buffer in the User's computer and inserted into the stream of regular programming upon receipt of a cueing signal embedded in the stream of regular programming. Thus, a substantial number of targeted groups, perhaps 10 or 100 or even more could be accommodated without an impractical increase in network load.

II. Control Architecture

The control architecture described in this section is responsible for managing and administering the Users who are receiving the information being delivered by the distribution architecture described in the previous section. The control architecture handles new User registration, User login, the starting and stopping of audio streams and the monitoring of ongoing transmissions. The control architecture is scalable just as is the distribution architecture so that any number of Users can be managed.

This section describes the control protocol, which consists of the format and sequence of control messages that are exchanged among Users, Control Servers, Media Servers, Primary Servers and the Administration Server. These messages are in the form of *objects*, which have specific data formats. Objects are exchanged preferably using the TCP protocol although other options are possible. Below we describe the sequence of objects passed among the various computers and detail the internal structure of each object.

The major objects used in the present embodiment of the invention are set forth in Table 1. For each object, Table 1 provides a brief description of its function, identification of the names of the fields in the object, their types and a brief description of their function.

TABLE 1

Channel Activation Object

Contains information used for channel activation/deactivation. It is sent to Media and Primary Servers to tell them to carry or stop carrying a specific channel. Media Servers get the channel from another server in the system hierarchy and Primary Servers get and encode the feed from the actual input source.

Field Name	Field Type	Remarks
Token	Security Token Object	
Moniker	Moniker Object	unique channel identifier
Activate	Int	action flag (activate/deactivate)
CompressType	Int	type of compression to use
Host	Host Object	host carrying the channel

Channel Guide Object

Contains analytical and descriptive information for an item requested that is uniquely identified by a moniker. It is usually the reply to a Channel Guide Request object.

Field Name	Field Type	Remarks
Token	Security Token Object	
Type	Int	type of content
Result		the content data itself

Channel Guide Request Object

Conveys a request for analytical and descriptive information about an item uniquely identified by the contained moniker. The reply is in the form of a Channel Guide object.

Field Name	Field Type	Remarks
Token	Security Token Object	inherited from base class
Type	Int	type of content
Moniker	Moniker Object	unique identifier

Table 1 (continued)

Host Object

Encapsulates the attributes of a networked computer related to the operation or services it offers or requests.

Field Name	Field Type	Remarks
Token	Security Token Object	
HostName	String	computer name and domain
PortNumber	Int	port number for service
DisplayName	String	descriptive computer name

Login Information Object

Encapsulates the name and password by which a User is known to the system.

Field Name	Field Type	Remarks
Token	Security Token Object	
Login	String	User's system login name
Password	String	User's system password (possibly encrypted)

Media Control Interface (MCI) Request Object

Encapsulates a multimedia control command, such as play and stop, and any extra information that may be necessary to perform the requested service.

Field Name	Field Type	Remarks
Token	Security Token Object	
Command	Int	multimedia command
String	String	command-specific extra info

Moniker Object

A moniker encapsulates the name of an object or process with the intelligence necessary to work with that name. In other words, it provides naming and binding services. The Moniker Object is used in the system for unique identification of various components, parts or features, such as a channel, a directory, or a computer list.

Field Name	Field Type	Remarks
Token	Security Token Object	
ID	String	unique string identifier
DisplayName	String	User-readable name

Table 1 (continued)

Ping Object

Ping is the name given to the "Are-You-Alive?" operation useful in determining if a specific computer is up and running. This object is used in the system when a server has to be queried for its operational status. It can also provide timing information for statistical purposes and quality of service evaluations.

Field Name	Field Type	Remarks
Token	Security Token Object	
Date	Date	system date
Time	Time	system time

Protocol List Object

Encapsulates a general purpose collection object.

Field Name	Field Type	Remarks
Token	Security Token Object	
Type	Int	type of object list

Result Message Object

Acts as the acknowledgment for a requested service successfully carried that out or reports errors that occur in the system during a client/server transaction.

Field Name	Field Type	Remarks
Token	Security Token Object	
Code	Int	result code
Message	String	message corresponding to code

Security Token Object

Contains the authorization key for a transaction. The key must be validated before any service is performed.

Field Name	Field Type	Remarks
ID	String	authorization key/transaction ID.

Table I (continued)

Server Activation Object

Contains information used in the server activation/deactivation process. Used for announcement as well as command purposes (e.g., a server can notify the administration database that is now activated or a server can be instructed to manage someone else).

Field Name	Field Type	Remarks
Token	Security Token Object	
Active	Int	action flag (activate/deactivate)
Manage	Int	control flag (manage/associate)
Type	Int	server type
Host	Host Object	host to be controlled

Server List Request Object

Encapsulates a request for a list of available server resources for an identified service (e.g., a request for a list of Control Servers for a specified channel).

Field Name	Field Type	Remarks
Token	Security Token Object	
Type	Int	type of service
Moniker	Moniker Object	content/channel unique identifier
Host	Host Object	local host information

Statistics Object

Contains system-related information that can be used by load-balancing algorithms and for statistical purposes.

Field Name	Field Type	Remarks
Token	Security Token Object	
Load	Int	load on the system
Threads	Int	number of threads running
Users	Int	number of Users being
Uptime	Int	serviced
NumberManaged	Int	amount of time running
NumberAssociated	Int	number of managed servers
		number of associated servers

Statistics Request Object

Encapsulates a request for system-related information that can be used by load-balancing algorithms and statistical purposes.

Field Name	Field Type	Remarks
Token	Security Token Object	
Load	Int	request flag (on/off)
Threads	Int	request flag (on/off)
Users	Int	request flag (on/off)
Uptime	Int	request flag (on/off)
NumberManaged	Int	request flag (on/off)
NumberAssociated	Int	request flag (on/off)

User Object

Users and Servers use this object to register themselves with the administration database. They provide the information for subsequent logins (name, password) and other system-related info. The end-Users provide personal, demographic, and system-related information.

Field Name	Field Type	Remarks
Token	Security Token Object	
Login	Login Information Object	login information(name, password)
FirstName	String	User's first name
LastName	String	User's last name
Title	String	User's job title
Company	String	User's employer
Address1	String	User's home street address
Address2	String	User's address extra
City	String	city, village
State	String	state, province or foreign country
ZipCode	String	zip or postal code
Age	String	User's age
Gender	String	User's gender
PhoneNumber	String	telephone number
FaxNumber	String	fax number
Email	String	email address
Demographics	Dictionary	market-targeting extra User info
SystemInfo	Dictionary	system-related information

Table 1 (continued)

Version Object

All components of the system use this object to report their versioning information to the party they transact with in order to use a protocol they both understand. They are also given the chance to update themselves if a newer version exists.

Field Name	Field Type	Remarks
Token	Security Token Object	
Major	Int	major protocol version number
Minor	Int	minor protocol version number
Type	Int	sender type
Client	Version	client version information

Unlike traditional protocols based on state computers, the control protocol of the present invention is a light-weight, stateless protocol comprising simple sequences of objects. It is *light-weight* in that in most sequences only two objects are involved in the transaction and after a sequence is completed the connection can be reused. It is also *stateless* in that the server maintains no information about the client. Every transaction is handled independently of the previous ones. States exist in the lower levels, for example within the TCP layer, to express logical states of a network connection but they are not actually part of the control protocol.

In the preferred embodiment, the software running on the Control Servers, Media Servers and Primary Servers is programmed for Windows NT and UNIX environment using the OLE environment. In addition, COM interfaces are used between components. The Rogue Wave system is used to transfer objects between the applications running on the various computers. The software running on the User computer is preferably programmed for a Windows 32-bit environment, so it will run on a Windows 95 or Windows NT computer. Alternatively, Macintosh and UNIX environments can be accommodated by other User software.

The basic process of a control transaction consists of a *version sequence* followed by one or more *protocol*

sequences. The version sequence starts after the computer initiating the transaction, the *client*, has established a connection with the computer completing the transaction, the *server*. The client sends a Version Object (defined in Table 1) and in response the server then sends back its own Version Object. This version sequence is used so that both client and server are aware of the version numbers of the software they are using. If a version number is older than expected, either client or server can choose to conform to the previous version or abort the transaction, depending on its needs and capabilities. If a version number is newer than expected, in most cases the current transaction can be completed since the software systems are designed to be fully backward compatible with previous versions. Additionally, in the case that the server of the transaction is the Administration Server, the client receives information about what the latest version number is and thus the client can be informed that a software update is needed. The process of handling automatic updating of User software is described more fully below.

After the version sequence, one or more protocol sequences occur in which other objects are exchanged between client and server. When a particular protocol sequence is completed, another independent protocol sequence can be serviced. The protocol sequences that are part of the control architecture of the present invention are summarized in Table 2 and described below in conjunction with Figures 6-17.

TABLE 2

Summary of Protocol Sequences

Control Sequence	Client	Server	Main Objects Exchanged
User Registration and Login (see Fig. 6)	User	Administration	Version Object User Object Channel Guide Object
User Login (see Fig. 7)	User	Administration	Version Object Login Information Object Channel Guide Object
Channel Play (see Figs 8a, 8B, 8C)	User	Administration	Version Object Server List Object
		Control	Version Object Server List Object
		Media	Version Object MCI Objects - OPEN/PLAY/STOP/CLOSE Ping Objects (TCP connection stays open)
Token Validation (see Figs. 9A, 9B)	Control or Media or Primary	Administration or Control	Version Object Security Token Object
Server Registration and Login (see Fig. 10)	Media or Control	Administration	Version Object User Object Server Activation Object
Server Login (see Fig. 11)	Media or Control	Administration	Version Object Login Object Server Activation Object
Control Server Activation (see Fig. 12)	Administration	Control	Version Object Server Activation Object

Control Sequence	Client	Server	Main Objects Exchanged
Media Server Activation (see Fig. 13)	Control	Media	Version Object Server Activation Object Ping Objects (TCP connection stays open)
Control Channel Activation (see Fig. 14)	Administration	Control	Version Object Channel Activation Object
Media Channel Activation (see Fig. 15)	Control	Media	(open TCP connection) Channel Activation Objects
Distribution Activation (see Fig. 16)	Media	Media or Primary	Version Object MCI Objects - OPEN/PLAY/STOP/CLOSE Ping Objects (TCP connection stays open)
Statistics Request (see Fig. 17)	Administration	Control or Media	Version Object Statistics Object

The User registration and login sequences are the processes by which a new User registers with the system, logs in and retrieves programming information. The channel play sequence takes place when a User asks to listen to a particular channel. The token validation sequence is used to verify that a computer requesting a service is authorized to do so. The Server registration, login and activation sequences are used by Control and Media Servers when they become active. The Control Server and Media Server activation sequences are used to manage the Control and Media Servers. The control channel, media channel and distribution activation sequences are used to cause a channel to be distributed to a Media Server. Finally, the statistics request is used for administrative purposes.

Fig. 6 illustrates the User registration and login sequence in more detail. This sequence takes place after the User has installed the User software on his/her computer. It is expected that the User will download the software from the Internet and then invoke it which in the preferred embodiment will use the Windows Wizard interface. This will guide the User through the installation process including filling out the registration form, which we will describe more fully in the next section. After the User has selected a name and password and selected the option to register, the User computer opens a TCP connection to the Administration Server. Advantageously, the full domain name of the Administration Server is embedded into the User software, although it could be discovered in other ways. The User and Administration Server then exchange version objects with the Administration Server as described above. If the version numbers meet expectations, the User sends a User Object to the Administration Server. The format of the User Object is shown in Table 1. Once the Administration Server receives the User Object, it verifies that the information is filled in properly and that the selected User name is unique. If the User Object is invalid for any reason, the Administration Server returns a Result Message Object with a code indicating the reason. The format of the Result Message Object is shown in Table 1. If the User information is valid, the Administration Server updates the global database of User names and passwords and then generates a security token for that User. This security token is then returned to the User in a Result Message Object.

Upon receiving the Result Message Object, the User saves the security token for future use. This token is an identifier that allows the User to request services from the Administration Server and other computers within the overall system. The security token is not saved permanently or registered on the User computer. Normally, the User software then immediately sends a Channel Guide Request Object to the Administration Server and a Channel Guide Object is returned.

The format of these objects is also shown in Table 1. Note that in principle, this is a separate transaction and could take place in a separate TCP connection to the Administration Server. In particular, once the User has registered and logged in, he/she can request the Channel Guide Object again since it may have been updated since the previous request. At this point the TCP connection to the Administration server is closed.

The process of User registration only needs to take place once for each User. However anyone can re-register at any time, even after the software has been installed. In particular, it is expected that if multiple persons use a computer, each person will register and obtain his/her own User name and password. If the registration process is not completed successfully, the User software saves the registration information and ask the User if they would like to try again the next time the software is invoked.

Since the security token is not permanently saved by the User software, it is lost when the User software is closed, and the security token must again be retrieved from the Administration Server the next time the User wants to use the system. This process is the purpose of the login sequence illustrated in Fig. 7. This sequence is used if a User has already registered and needs only to retrieve a valid security token. In this case the sequence consists of the User's sending a Login Information Object to the Administration Server. The Administration Server then queries the User database to validate the login name and password. If the login name and password are correct, then a security token is returned to the User. Normally the receipt of the security token will immediately be followed by a channel information request sequence, just as in the registration sequence described previously.

The control sequence that takes place when a User initiates a channel play operation is illustrated in Figs. 8A, 8B and 8C. First the User software requests a Control Server List from the Administration Server. Note that the

Server List Request Object, illustrated in Table 1 contains a channel identifier. The Administration Server generates a sorted list of Control Servers based on overall system load and the location of the User on the network and returns this list to the User using a Protocol List Object. Once the Control Server List is returned to the User, the Administration Server is no longer needed and the TCP connection is closed.

The User software then searches the list of Control Servers and opens a TCP connection to the first host listed. If that host computer does not respond, then the next Control Server on the list is tested and so forth in succession. Upon obtaining a response from a Control Server, the User software uses a Server List Request Object to requests a Media Server List from the Control Server. If the Control Server is too busy to service the User, it returns a Result Message Object so indicating and the User software tries the next Control Server on the list. However, in the likely scenario that the Control Server is able to handle the User's request, a sorted list of Media Servers is generated and returned to the User computer using a Protocol List Object. The TCP connection to the Control Server is then closed by the User software.

At this point the User software initiates a TCP connection to the first Media Server on the list provided by the Control Server. As in the previous case, it attempts to connect to the first host on the list and if unsuccessful tries the next hosts in succession. Once the Version Objects are exchanged, the User software sends an MCI Request Object to the Media Server. An MCI Request Object can be used for four basic commands: OPEN, PLAY, STOP and CLOSE. The User software must first send an OPEN command for the desired channel. If the returned Result Message Object indicates success, the User software then sends a PLAY command.

When the Media Server receives a valid PLAY command, it initiates the delivery of audio information to the User as described in the previous section. Note that

this could be in the form of broadcast, multicast or unicast packets to a specific UDP port. The TCP connection through which the MCI Request Objects were sent stays open during the audio play operation. In addition, Ping Objects are sent to the User on a periodic basis to verify that the computer is still working and active. When the User software receives a Ping Object, it simply returns it. The Media Server uses the Ping Objects to measure round trip time and also to determine when a User's computer has terminated abnormally. In that case the audio stream is terminated.

In the case of normal termination of the audio stream, the User makes an explicit selection to stop and this causes a STOP command to be sent to the Media Server in an MCI Request Object. The Media Server then terminates the audio stream to that User. When the User closes the application software or selects another channel to play, the User software will send a CLOSE command to the Media Server in an MCI Request Object and the TCP connection is closed.

The initiation of the audio stream by the Media Server causes a log entry to be generated and sent to the Administration Server. This information is important so that the Administration Server can update its database to indicate which Users are listening to which channels. The security token is used to identify the User initiating the audio stream. Additionally, when the audio stream is terminated to any User, another log message is generated and sent to the Administration Server.

Fig. 9A illustrates the process by which security tokens are validated. The Administration Server is the only server that can validate a security token. Thus, when a User requests services from a Control Server or from a Media Server, that server must go back to the Administration Server with a token validation sequence. However, Control Servers and Media Servers are allowed to cache validations of security tokens so that they do not have to validate tokens repeatedly once they have validated it the first time. In the case where a Media Server receives a request, the token

will be validated with the Control Server that is managing that Media Server. Fig. 9B identifies the various token validation scenarios.

Fig. 10 illustrates the process by which a new Server is registered. This process is similar to new User registration. It is expected, however, that the server installation will be through a Web interface rather than a Wizard. The Administration Server, upon receiving a User Object from a Media Server or Control Server validates the User name and password and generate a security token just as in the case of User registration. Normally the Server then immediately sends back a Server Activation Object indicating that it is ready to be used as a system resource. Once this process has been completed, the TCP connection to the Administration Server is closed.

If a Media Server or Control Server that has sent a Server Activation Object to the Administration Server becomes inactive, it will send another Server Activation Object indicating this condition. In the case of a Media Server, this object is sent to the managing Control Server. In the case of a Control Server, this object sent to the Administration Server. As in the case of User registration, Media Server and Control Server registration needs only take place once per computer. However, if the computer is restarted, the server must login and again retrieve a security token. This is the server login and activation sequence shown in Figure 11.

Once a Control Server has indicated to the Administration Server that it is ready, the Administration Server can activate that Control Server by sending the Control Server a Server Activation Object as illustrated in Fig. 12. This is a separate transaction and is used to tell the Control Server which Media Servers it is supposed to manage. Recall that a Control Server and a number of Media Servers form a *cluster* of Media Servers. The single Control Server that manages that cluster must be given a list of host computers corresponding to the Media Servers in that cluster.

The process by which a Control Server activates the Media Servers that it manages is illustrated in Fig. 13. The Control Server sends a Server Activation Object to the Media Server indicating that it is responsible for channel management. This TCP connection between the Control Server and the Media Server stays open during the time that both servers are active. The Control Server periodically sends Ping Objects to the Media Server across this open TCP connection to verify that the Media Server is still running.

Fig. 14 illustrates the process by which a given channel is activated by the Administration Server. The Administration Server opens a connection to a Control Server that its wishes to have carry a given channel and provide a Channel Activation Object. This object indicates to the Control Server which Media or Primary Server the Control Server should direct its Media Servers to get the feed from. At this point the Control Server is said to be carrying that channel and it will be a valid host on a list of Control Servers requested by a Channel Play sequence.

Fig. 15 illustrates what happens when a Control Server needs to provide a channel. First it sends a Channel Activation Object to one of the Media Servers that it manages across the open TCP connection described previously. This object indicates to the Media Server that it should start receiving the channel identified and from where it should receive it.

In Figs. 16A and 16B depict how the Media Server requests distribution of an audio channel from another Media Server or from a Primary Server. This sequence is much the same as that in which a User requests the distribution of audio information from a Media Server. Note that a Media Server receives a single incoming stream for each channel that it is carrying and will then redistributes this stream to all Users or other Media Servers that request it.

Finally, Fig. 17 illustrates the statistics request sequence. This sequence is used by the Administration Server to gather information from the Media Servers and Control

Servers in order to manage the overall system. It can use this information to detect failures and to balance load as the dynamic conditions change. As indicated above, it can also use this information to monitor which Users are listening to which channel or whether Users stop listening to a channel at any time, such as during the play of a particular song. It can also use this information to control the advertising content that is downloaded to a particular User in advance of receipt of regular audio programming and/or monitor the delivery of advertising to the Users.

The control architecture described in this section is scalable to handle any number of Users. Note that the User registration process only happens once for each subscriber and the login process only happens once per session. These interactions, which require the Administration Server are expected to constitute a very small percentage of the overall system bandwidth. If the Administration Server were to become a bottleneck, however, it would be possible to duplicate it and to have the database it maintains distributed and automatically updated to guarantee consistency.

The Control Servers are distributed throughout the network and can handle the lower level interactions with the Users and the Media Servers. A single Control Server can handle preferably on the order of ten Media Servers up to several hundred Users. The bitrate among the Users, the Control Servers and the Media Servers is expected to be small in comparison to the audio transmission bitrate. The Ping Objects normally only involve the User and the nearest Media Server. They are also low in overhead since they are small and only get transmitted infrequently.

III. User Interface

The User interface is provided by the client application running on an individual computer and its associated graphical interface. In the preferred embodiment the User interface is available for 32-bit Windows (95 and NT),

Macintosh and UNIX platforms. Preferably anyone on the Internet can freely download a copy of the client software and install it in their computer.

Figure 18 illustrates the main User screen in the preferred embodiment. The screen is composed of three sections: channel guide (upper left frame), program guide (upper right frame), and multimedia frame (lower half of screen). The channel guide lists, as a tree hierarchy, the channels that are available from the system. The User selects a channel from the list of those displayed on the channel guide. The program guide provides information pertaining to the channel selected. This information can be a detailed schedule of the programming that has played or will be playing on the channel selected. Additionally, other relevant information will be displayed in this frame, for example, a notice regarding an upcoming special event on another channel. The multimedia frame provides an integrated web browser that displays information via a series of tabbed sections.

The information contained in the channel guide, program guide, and the tabs of the multimedia frame is dynamically transmitted to the client. For example, if a new channel begins operation, the client application can immediately display it as being available. Furthermore, the tabs displayed can be specifically relevant depending on what song is playing. For example, tabs displaying the album cover, information on the artist, song lyrics, tour dates can be displayed. Additionally, as shown in the example in figure 18, a tab can be available allowing the User to place an order for the CD or allowing the User to participate in a chat session related to the channel.

Figure 19 illustrates the key pull-down menus available in the main User screen in the preferred embodiment. Table 3 provides a description of each of the functions available through the pull down menus, as shown in figure 19.

As will be apparent to those skilled in the art, numerous modifications may be made within the spirit and scope of the invention.

Table 3

Pull-Down Menu Functions

Menu Choice	Menu Sub-Choice	Description
File	Login	Allows the User to login to the system.
	Logout	Allows the User to logout from the system.
	Register	Brings up a dialog so that the User can register with the system for the first time.
	Close	Minimizes the screen.
Edit	Copy	Allows the User to copy the selection on to the clipboard.
	Properties	Allows the User to set various properties.
Audio	Play	Begins playing the selected channel.
	Stop	Stops playing the selected channel.
	Mute	Stops the playing of audio
View	Tool Bar	Display or hide the tool bar (providing access to pull-down menu functions).
	Status Bar	Display or hide the status bar normally situated at bottom of the screen.
	Web Bar	Display or hide the tool bar section that provides access to the web browser functions.
Help	Help Topics	Brings up a list of available online help topics.
	About...	Displays summary infirmation regarding this application, such as version number, copyright information, and so on.

What is claimed is:

1. A method for transmitting message packets over a communications network comprising the steps of:

converting a plurality of streams of audio and/or visual information into a plurality of streams of addressed digital packets complying with the specifications of a network communication protocol,

for each stream, routing such stream to one or more users,

controlling the routing of the stream of packets in response to selection signals received from the users, and

monitoring the reception of packets by the users and accumulating records which indicate which streams of packets were received by which users.

2. The method of claim 1 further comprising the step of including in at least one stream of packets at least some advertising information.

3. The method of claim 2 further comprising the step of varying the content of the advertising information with the identity of the user to whom the advertising information is provided.

4. The method of claim 2 wherein the advertising information is inserted into the stream of audio and/or visual information before such stream is converted into a stream of packets.

5. The method of claim 1 further comprising the step of generating an audio output and/or a visual display from the stream of packets received by the user.

6. The method of claim 1 further comprising the steps of:

storing a first stream of packets received by the user at a first time and

at a later time, inserting the first stream of packets into a second stream of packets received at the user.

7. The method of claim 6 further comprising the step of converting the combined first and second streams of packets into an audio output and/or visual display.

8. The method of claim 6 wherein the first stream of packets contains advertising information.

9. The method of claim 6 wherein the content of the advertising information is varied depending on the identity of the user.

10. The method of claim 2 wherein the records that are accumulated indicate how many users received specific advertising information.

11. The method of claim 1 wherein at least one stream of packets comprises copyrighted music selections and the records that are accumulated indicate how many users received specific music selections.

12. The method of claim 1 wherein at least one stream of packets comprise music selections and the records that are accumulated indicate how many users did or did not listen to the entire selection.

13. The method of claim 1 further comprising the steps of:

compressing the stream of packets in their passage from source to user, and

decompressing the stream of packets near the user.

14. The method of claim 3 where in the compressing step uses a compression algorithm that is selected in accordance

with the content of the information being communicated in the stream of packets.

15. The method of claim 4 wherein the compressing step inserts into each packet an identification of the compression algorithm used and the decompressing step monitors each packet to read such identification and to vary its decompression algorithm in response thereto.

16. A method for transmitting at least one stream of audio and/or visual information over a communications network to a plurality of users comprising the steps of:

controlling the routing of the stream of information through the network in response to selection signals received from the users, and

monitoring the reception of the stream of information by the users and accumulating records relating to the reception of the stream of information by the users.

17. The method of claim 16 further comprising the step of including in at least one stream of information at least some advertising information.

18. The method of claim 17 further comprising the step of varying the content of the advertising information with the identity of the user to whom the advertising information is provided.

19. The method of claim 16 further comprising the steps of:

storing a first stream of information received by the user at a first time and

at a later time, inserting the first stream of information into a second stream of information received by the user.

20. The method of claim 19 wherein the first stream of information contains advertising information.

21. The method of claim 17 wherein the records that are accumulated indicate how many users received specific advertising information.

22. The method of claim 17 wherein at least one stream of packets comprises copyrighted music selections and the records that are accumulated indicate how many users received specific music selections.

23. The method of claim 17 wherein at least one stream of packets comprise music selections and the records that are accumulated indicate how many users did or did not listen to the entire selection.

24. The method of claim 17 further comprising the steps of:

compressing the stream of information in its passage from source to user, and

decompressing the stream of information near the user.

25. The method of claim 24 wherein the compressing step uses a compression algorithm that is selected in accordance with the content of the information being communicated in the stream of information.

26. The method of claim 16 wherein multiple streams of audio and/or visual information are transmitted over the communications network and the user can select which stream to receive.

27. A communication system comprising:

means for converting at least one stream of audio and/or visual information into a stream of addressed digital

packets complying with the specifications of a network communication protocol,

means for routing such stream via a communication network to selected users,

means for controlling the routing of the stream of packets in response to selection signals received from the users, and

means for monitoring the reception of packets by the user and for accumulating records which indicate which streams of packets were received by which users.

28. The communication system of claim 27 further comprising means for including in the stream of packets at least some advertising information.

29. The communication system of claim 28 further comprising means for varying the content of the advertising information with the identity of the user to whom the advertising information is provided.

30. The communication system of claim 27 further comprising means for generating from the stream of packets received at the user an audio output and/or a visual display.

31. The communication system of claim 27 further comprising means for storing packets received at the user during a first time period and means for inserting such packets into other packets received at the user at a later time period.

32. The communication system of claim 31 wherein the stream of packets received during the first time period contains advertising information.

33. The communication system of claim 32 wherein the content of the advertising information is varied depending on the identity of the user.

34. The communication system of claim 27 further comprising:

means for compressing the stream of packets in their passage from source to user, and

downstream of the compressing means, means for decompressing the stream of packets.

35. The communication system of claim 34 wherein the compressing means is located near the converting means and the decompressing means is located at the user.

36. The communication system of claim 34 wherein the compressing means uses a compression algorithm that is selected in accordance with the content of the information being communicated in the stream of packets.

37. The communication system of claim 34 wherein the compressing means inserts into each packet an identification of the compression algorithm used and the decompressing means monitors each packet to read such identification and to vary its decompression algorithm in response thereto.

FIGURE 1

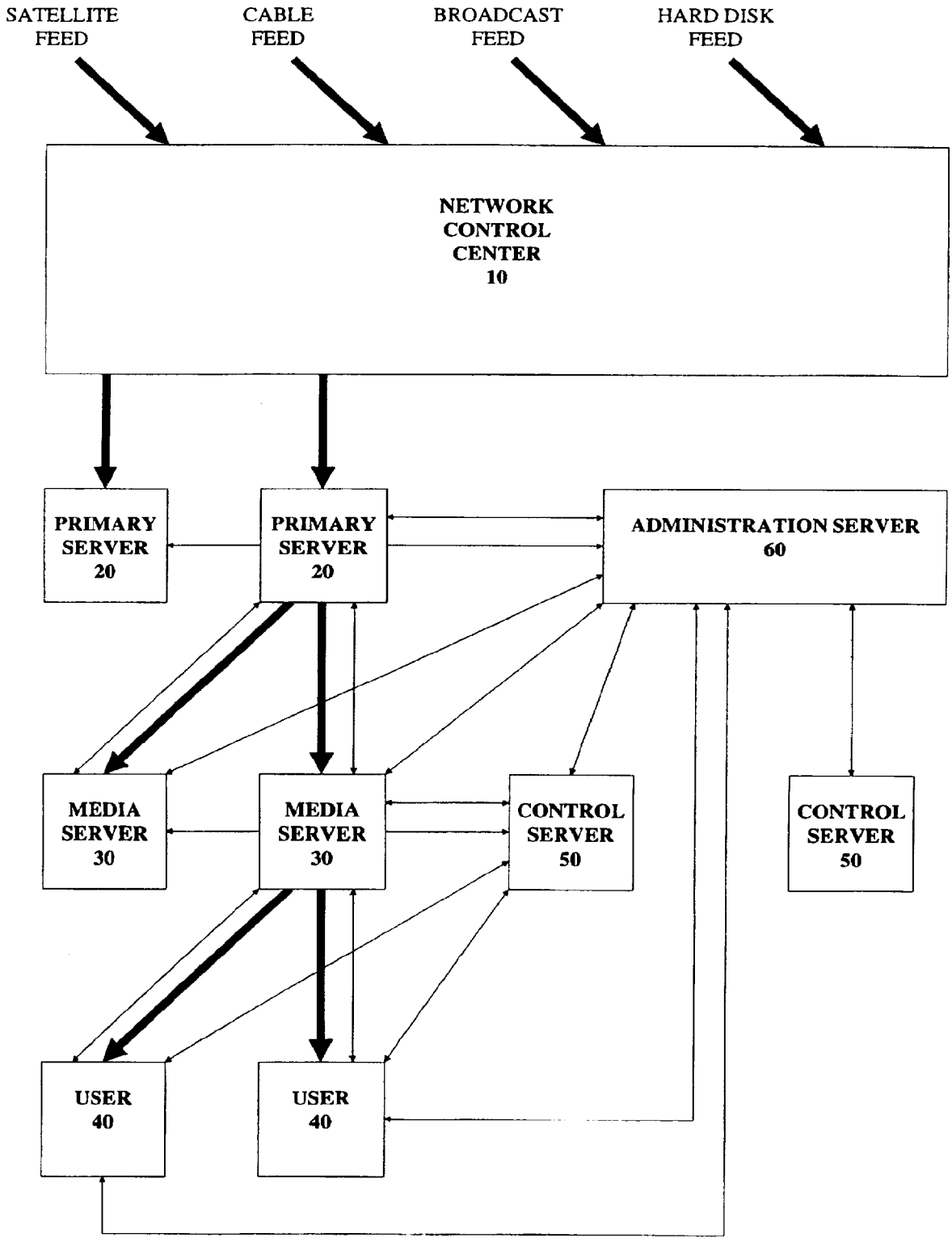

SATELLITE FEED
CABLE FEED
BROADCAST FEED
HARD DISK FEED
NETWORK CONTROL CENTER 10
PRIMARY SERVER 20
PRIMARY SERVER 20
ADMINISTRATION SERVER 60
MEDIA SERVER 30
MEDIA SERVER 30
CONTROL SERVER 50
CONTROL SERVER 50
USER 40
USER 40

FIGURE 2

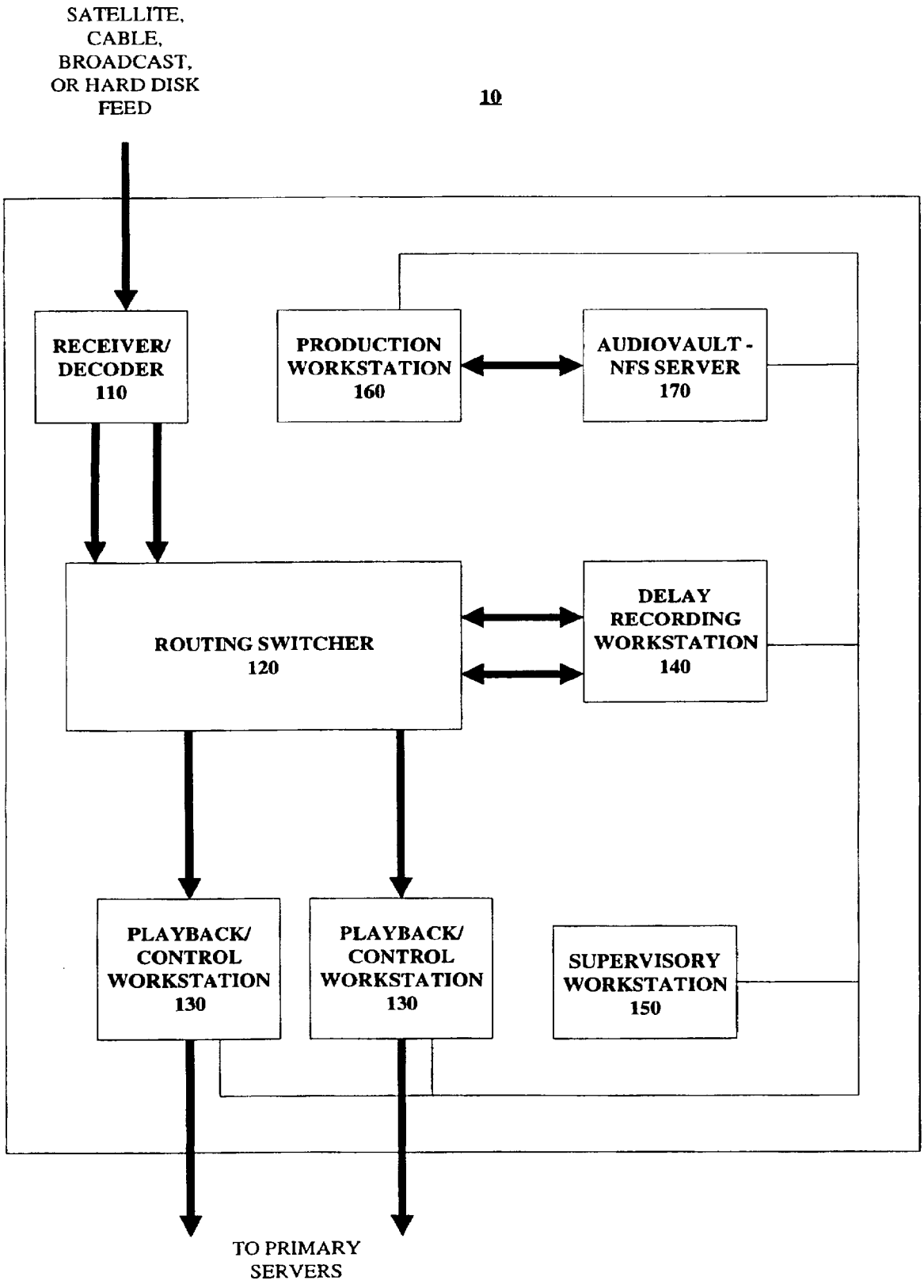

SATELLITE, CABLE, BROADCAST, OR HARD DISK FEED
10
RECEIVER/ DECODER 110
PRODUCTION WORKSTATION 160
AUDIOVAULT - NFS SERVER 170
ROUTING SWITCHER 120
DELAY RECORDING WORKSTATION 140
PLAYBACK/ CONTROL WORKSTATION 130
PLAYBACK/ CONTROL WORKSTATION 130
SUPERVISORY WORKSTATION 150
TO PRIMARY SERVERS

FIGURE 3

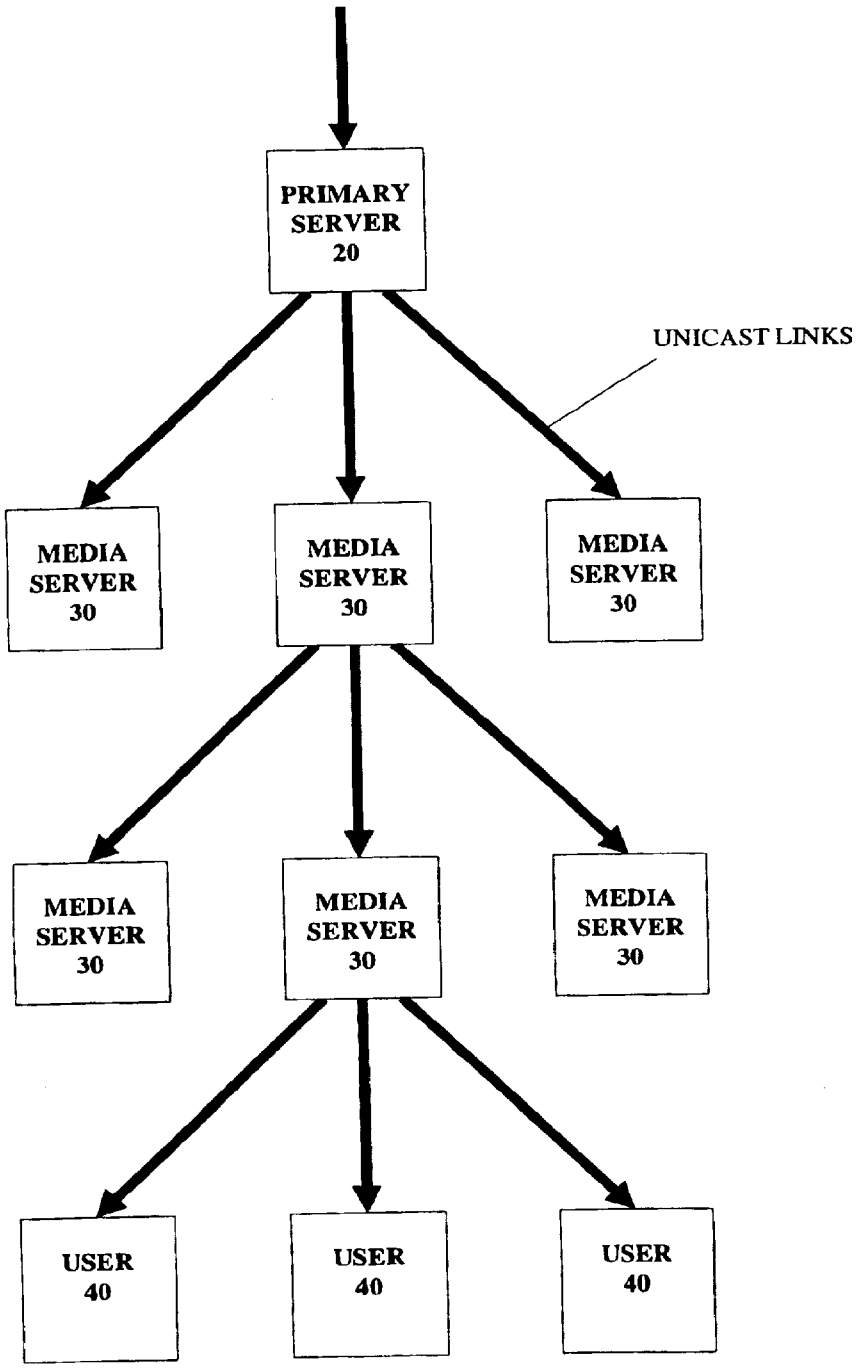

PRIMARY SERVER 20
UNICAST LINKS
MEDIA SERVER 30
MEDIA SERVER 30
MEDIA SERVER 30
MEDIA SERVER 30
MEDIA SERVER 30
MEDIA SERVER 30
USER 40
USER 40
USER 40

FIGURE 4

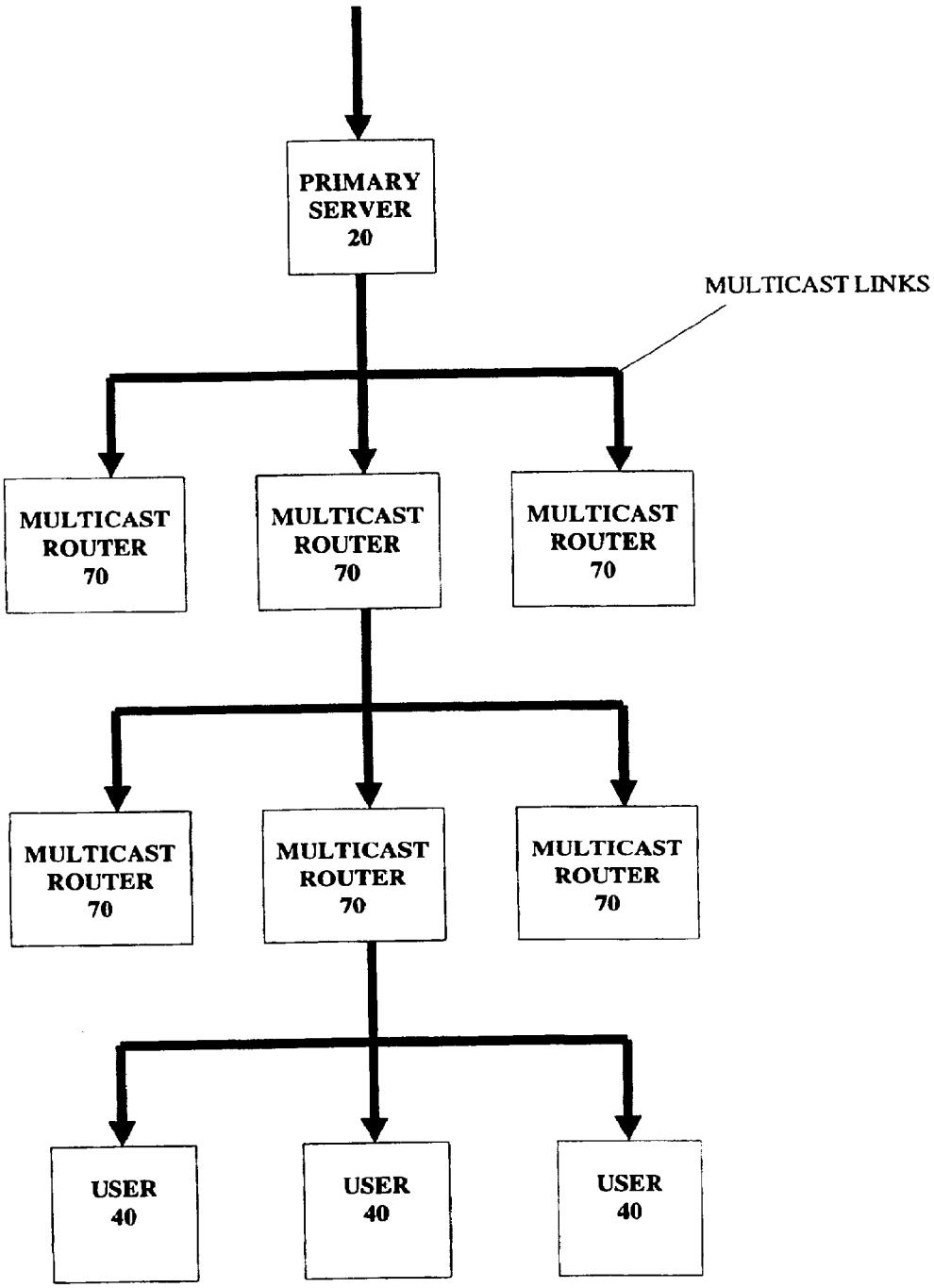

PRIMARY SERVER 20
MULTICAST LINKS
MULTICAST ROUTER 70
MULTICAST ROUTER 70
MULTICAST ROUTER 70
MULTICAST ROUTER 70
MULTICAST ROUTER 70
MULTICAST ROUTER 70
USER 40
USER 40
USER 40

FIGURE 5

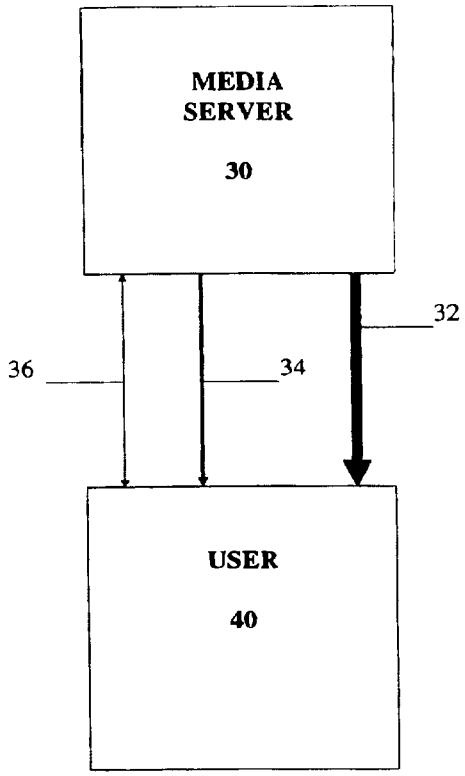


FIGURE 6

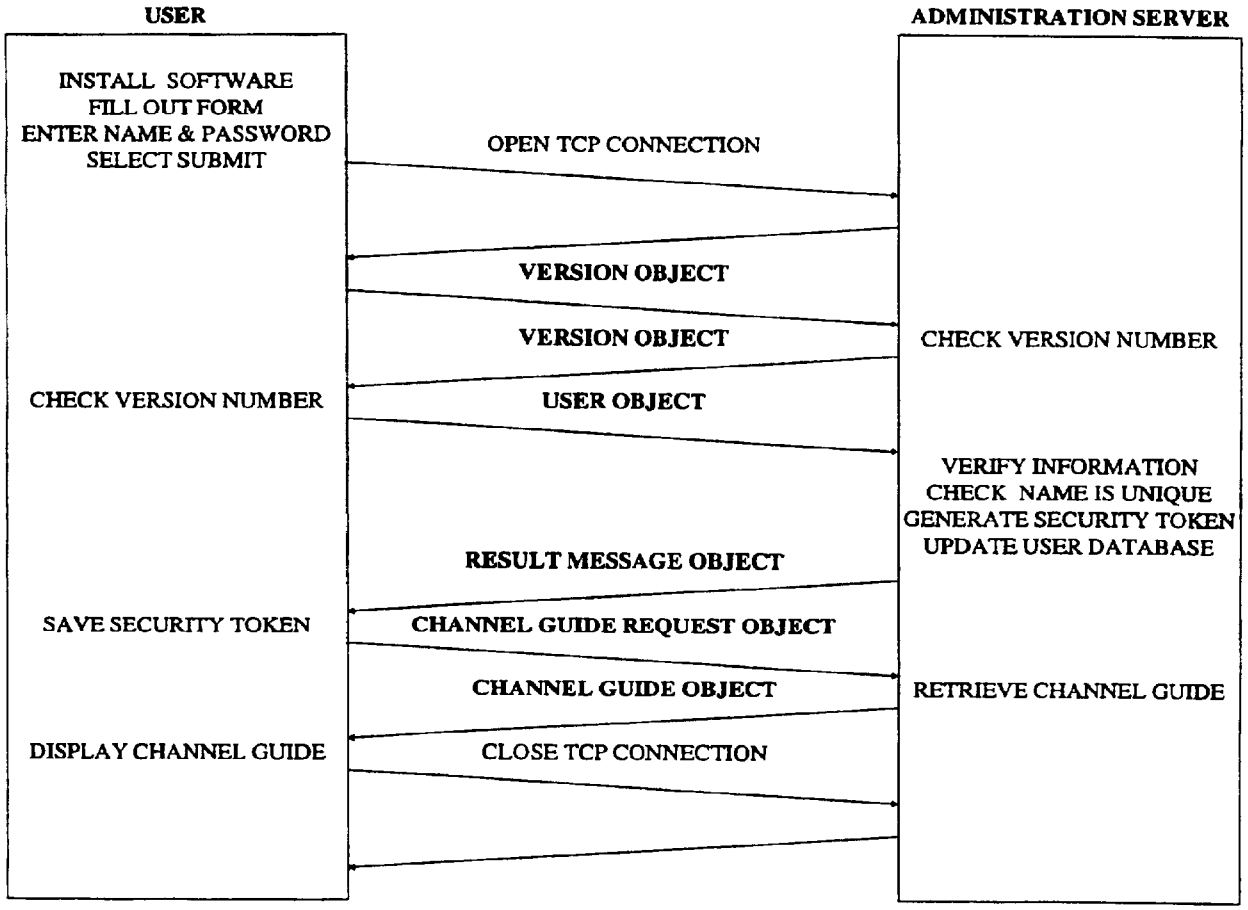

USER
ADMINISTRATION SERVER
INSTALL SOFTWARE
FILL OUT FORM
ENTER NAME & PASSWORD
SELECT SUBMIT
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
USER OBJECT
VERIFY INFORMATION
CHECK NAME IS UNIQUE
GENERATE SECURITY TOKEN
UPDATE USER DATABASE
RESULT MESSAGE OBJECT
SAVE SECURITY TOKEN
CHANNEL GUIDE REQUEST OBJECT
CHANNEL GUIDE OBJECT
RETRIEVE CHANNEL GUIDE
DISPLAY CHANNEL GUIDE
CLOSE TCP CONNECTION

FIGURE 7

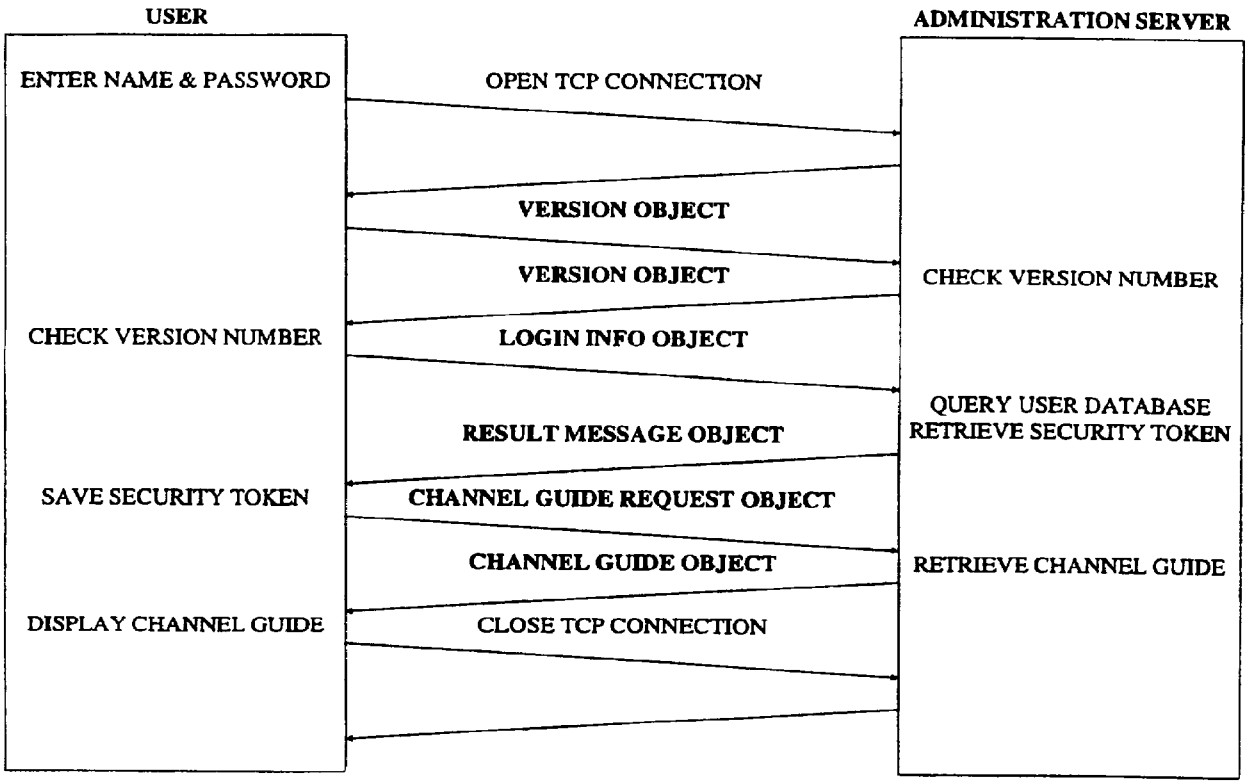

USER
ADMINISTRATION SERVER
ENTER NAME & PASSWORD
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
LOGIN INFO OBJECT
QUERY USER DATABASE
RETRIEVE SECURITY TOKEN
RESULT MESSAGE OBJECT
SAVE SECURITY TOKEN
CHANNEL GUIDE REQUEST OBJECT
CHANNEL GUIDE OBJECT
RETRIEVE CHANNEL GUIDE
DISPLAY CHANNEL GUIDE
CLOSE TCP CONNECTION

FIGURE 8A

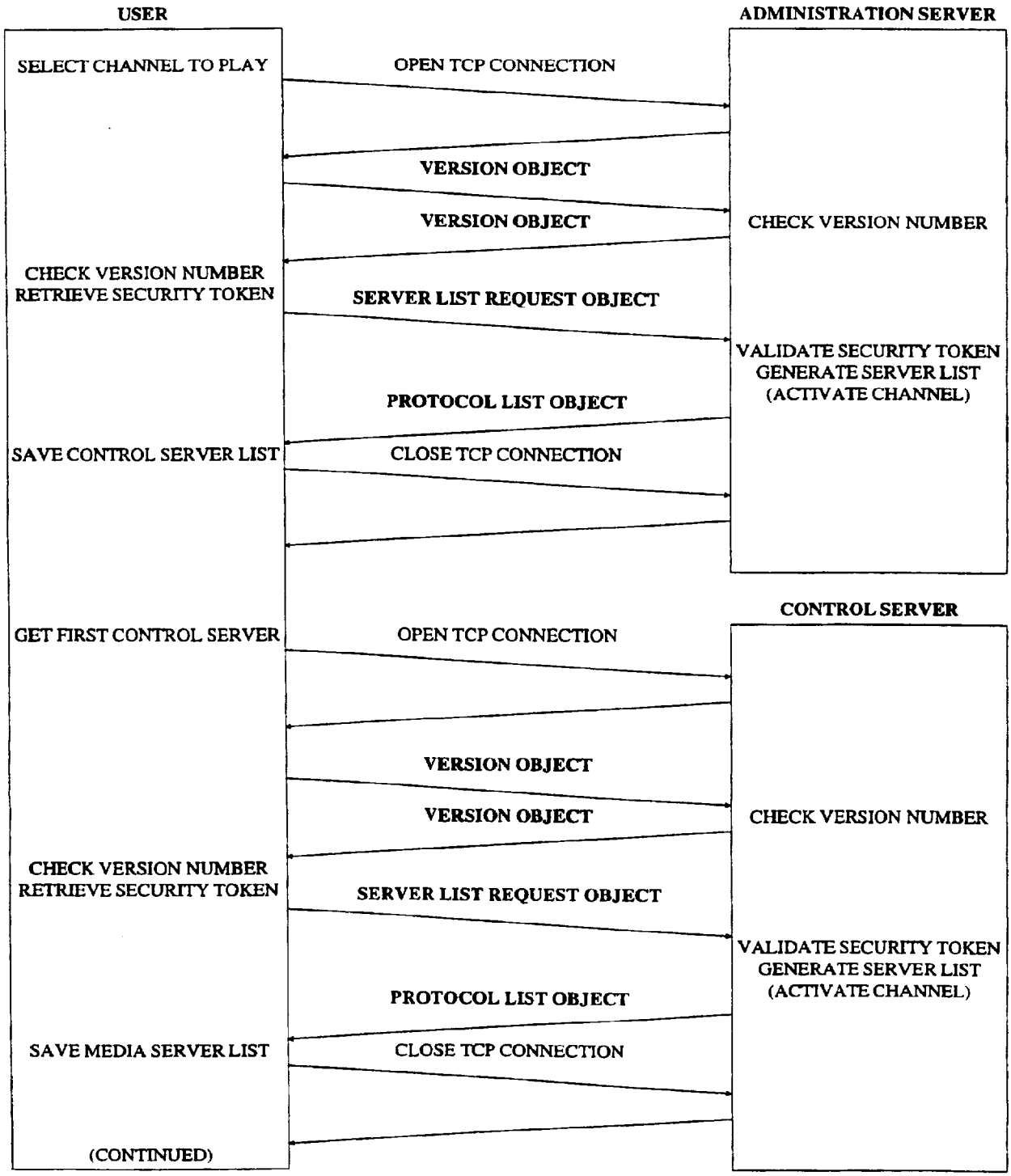

USER
ADMINISTRATION SERVER
SELECT CHANNEL TO PLAY
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
RETRIEVE SECURITY TOKEN
SERVER LIST REQUEST OBJECT
VALIDATE SECURITY TOKEN
GENERATE SERVER LIST
(ACTIVATE CHANNEL)
PROTOCOL LIST OBJECT
SAVE CONTROL SERVER LIST
CLOSE TCP CONNECTION
CONTROL SERVER
GET FIRST CONTROL SERVER
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
RETRIEVE SECURITY TOKEN
SERVER LIST REQUEST OBJECT
VALIDATE SECURITY TOKEN
GENERATE SERVER LIST
(ACTIVATE CHANNEL)
PROTOCOL LIST OBJECT
SAVE MEDIA SERVER LIST
CLOSE TCP CONNECTION
(CONTINUED)

FIGURE 8B

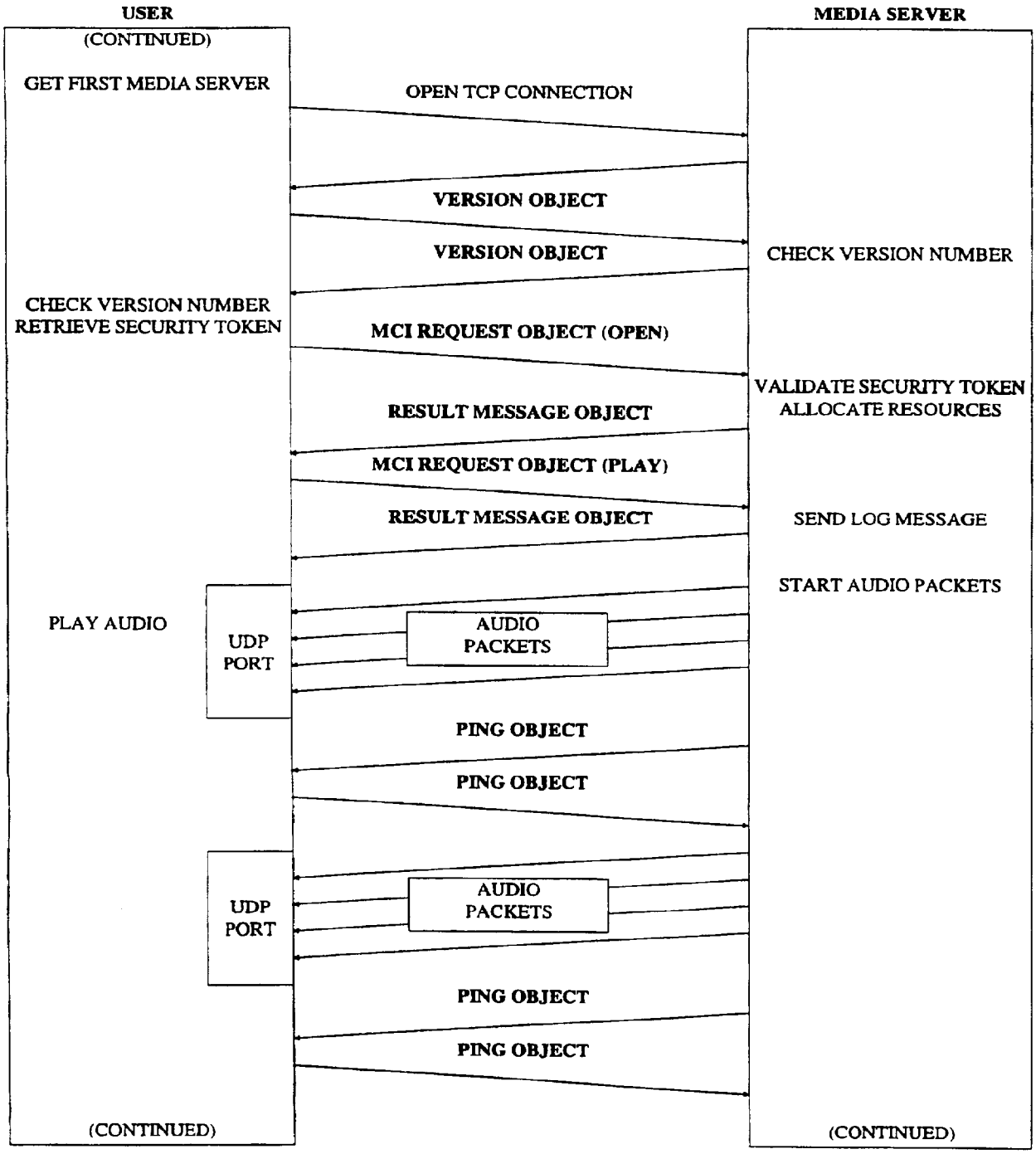

USER
(CONTINUED)
MEDIA SERVER
GET FIRST MEDIA SERVER
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
RETRIEVE SECURITY TOKEN
MCI REQUEST OBJECT (OPEN)
VALIDATE SECURITY TOKEN
ALLOCATE RESOURCES
RESULT MESSAGE OBJECT
MCI REQUEST OBJECT (PLAY)
RESULT MESSAGE OBJECT
SEND LOG MESSAGE
START AUDIO PACKETS
PLAY AUDIO
UDP PORT
AUDIO PACKETS
PING OBJECT
PING OBJECT
UDP PORT
AUDIO PACKETS
PING OBJECT
PING OBJECT
(CONTINUED)
(CONTINUED)

FIGURE 8C

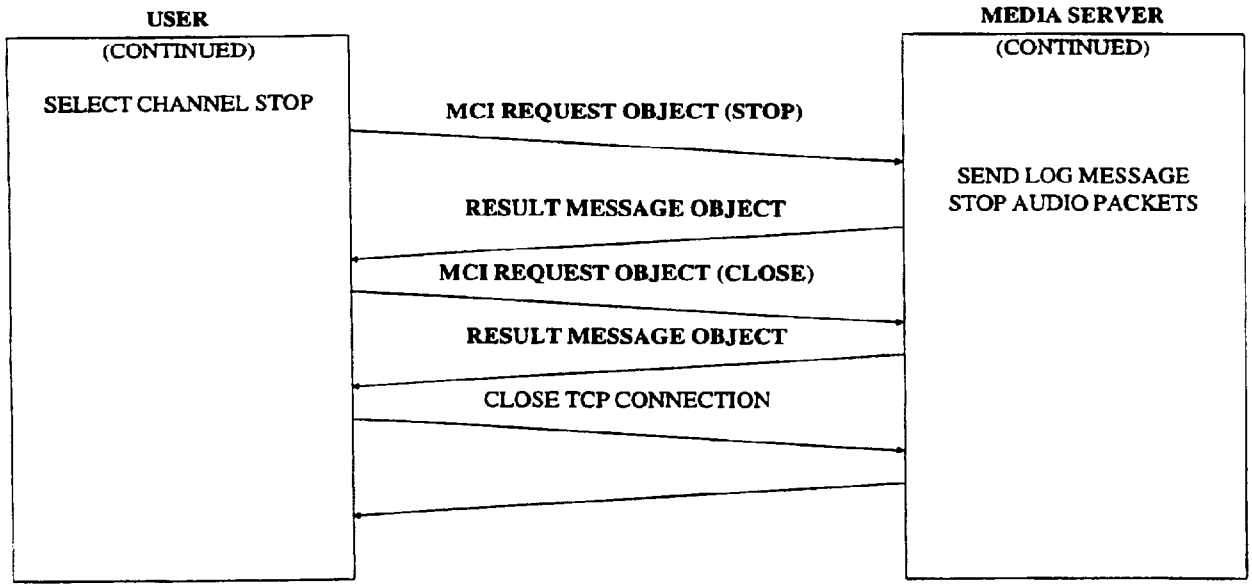

USER
(CONTINUED)
SELECT CHANNEL STOP
MEDIA SERVER
(CONTINUED)
MCI REQUEST OBJECT (STOP)
SEND LOG MESSAGE
STOP AUDIO PACKETS
RESULT MESSAGE OBJECT
MCI REQUEST OBJECT (CLOSE)
RESULT MESSAGE OBJECT
CLOSE TCP CONNECTION

FIGURE 9A

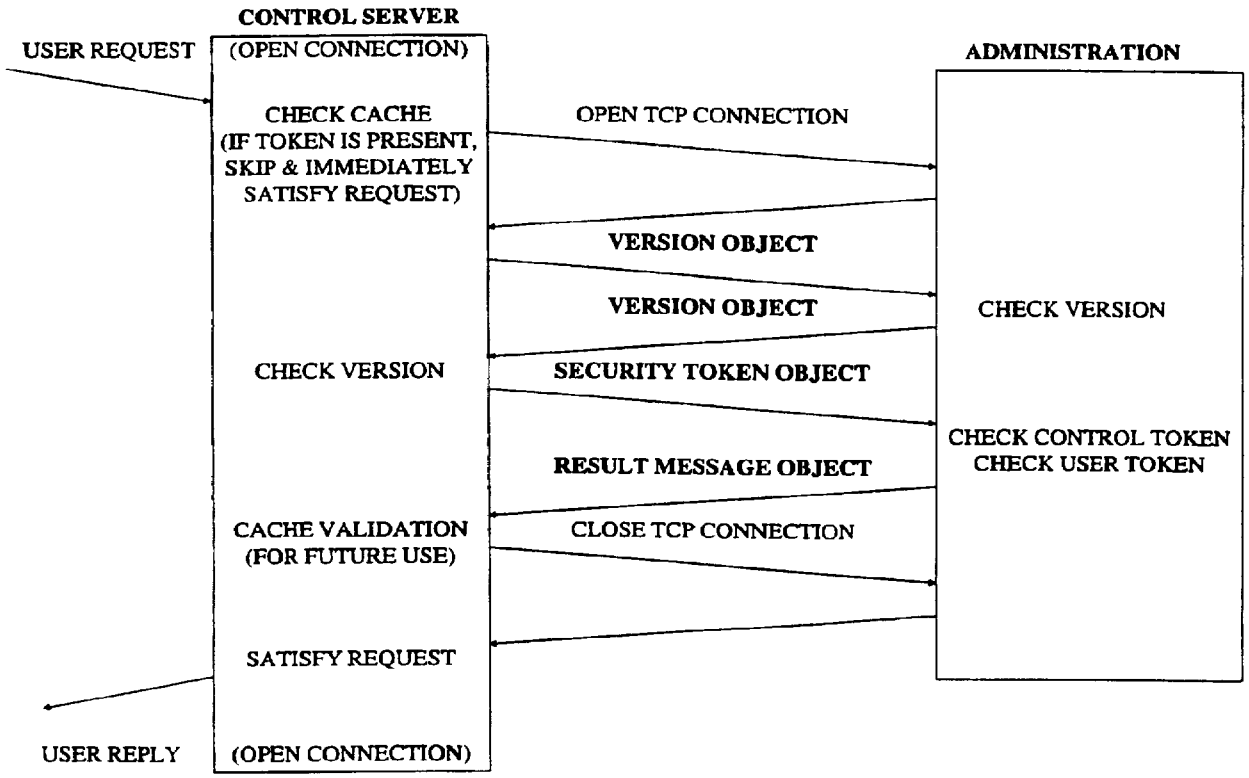

CONTROL SERVER
USER REQUEST
(OPEN CONNECTION)
ADMINISTRATION
CHECK CACHE
(IF TOKEN IS PRESENT,
SKIP & IMMEDIATELY
SATISFY REQUEST)
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION
CHECK VERSION
SECURITY TOKEN OBJECT
CHECK CONTROL TOKEN
CHECK USER TOKEN
RESULT MESSAGE OBJECT
CACHE VALIDATION
(FOR FUTURE USE)
CLOSE TCP CONNECTION
SATISFY REQUEST
USER REPLY
(OPEN CONNECTION)

FIGURE 9B

(SHOWN ABOVE)

REQUEST FROM	REQUEST TO	VALIDATION WITH
USER	CONTROL SERVER	ADMINISTRATION SERVER
USER	MEDIA SERVER	CONTROL SERVER
MEDIA SERVER	MEDIA SERVER	CONTROL SERVER
MEDIA SERVER	PRIMARY SERVER	ADMINISTRATION SERVER
MEDIA SERVER	CONTROL SERVER	ADMINISTRATION SERVER
CONTROL SERVER	MEDIA SERVER	ADMINISTRATION SERVER

FIGURE 10

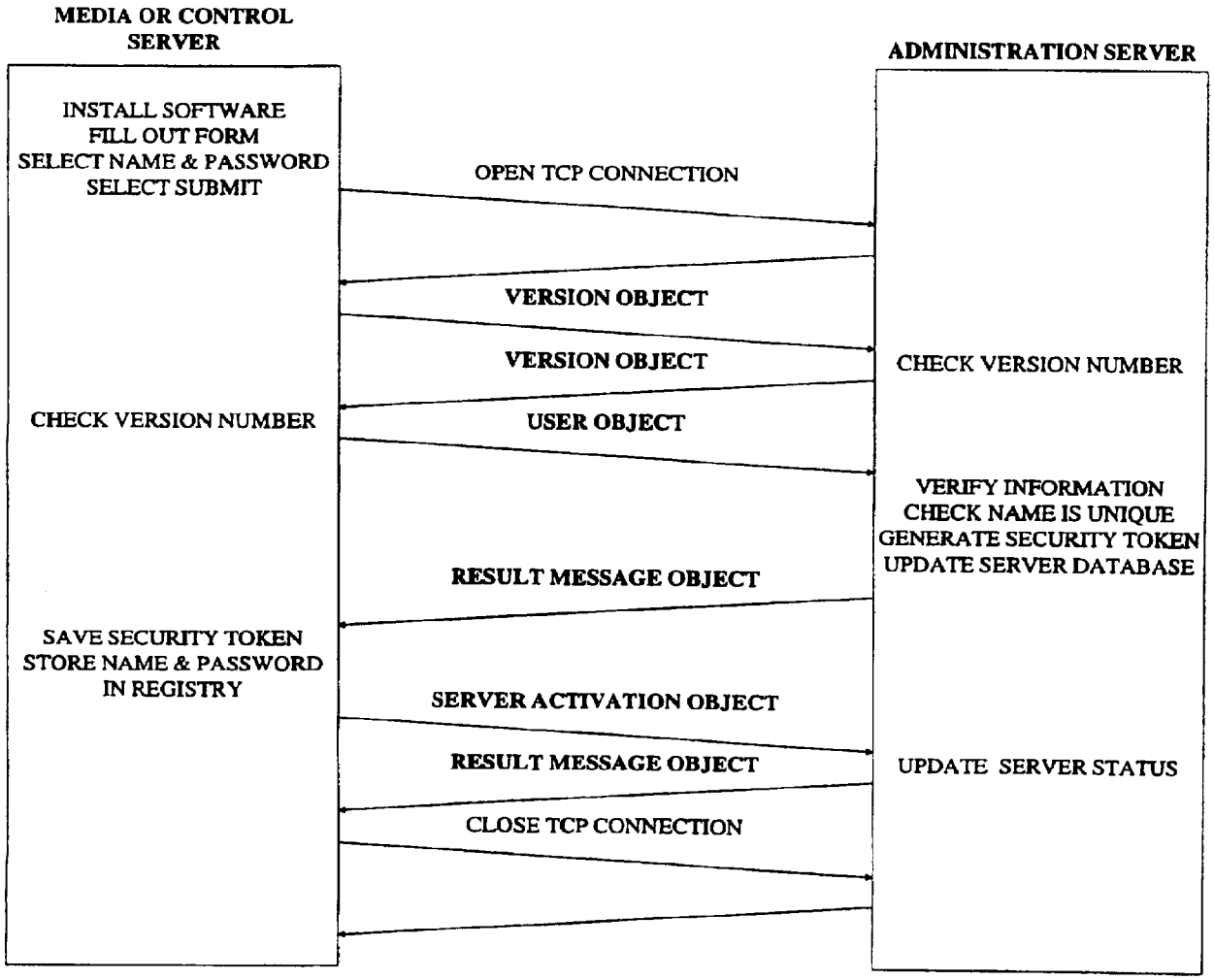

MEDIA OR CONTROL SERVER
ADMINISTRATION SERVER
INSTALL SOFTWARE
FILL OUT FORM
SELECT NAME & PASSWORD
SELECT SUBMIT
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
USER OBJECT
VERIFY INFORMATION
CHECK NAME IS UNIQUE
GENERATE SECURITY TOKEN
UPDATE SERVER DATABASE
RESULT MESSAGE OBJECT
SAVE SECURITY TOKEN
STORE NAME & PASSWORD
IN REGISTRY
SERVER ACTIVATION OBJECT
RESULT MESSAGE OBJECT
UPDATE SERVER STATUS
CLOSE TCP CONNECTION

FIGURE 11

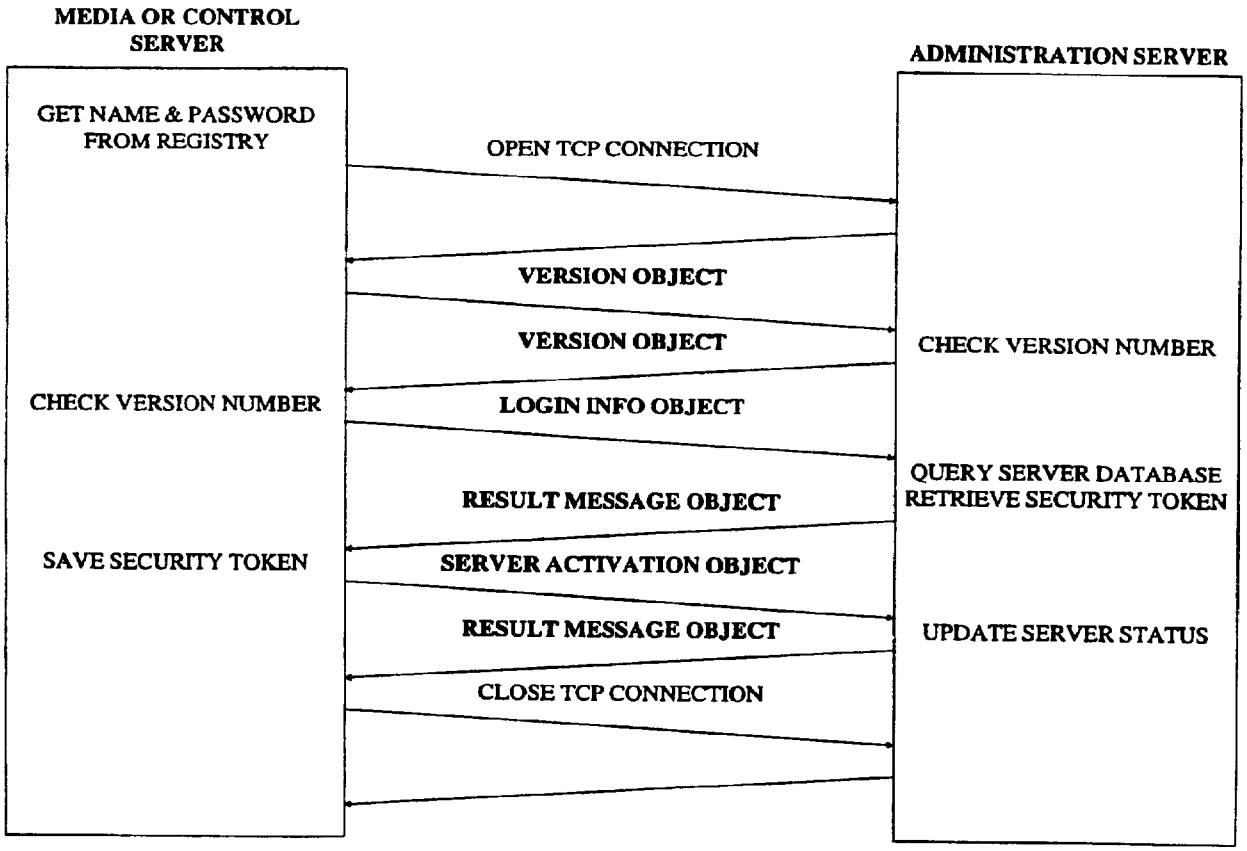

MEDIA OR CONTROL SERVER
ADMINISTRATION SERVER
GET NAME & PASSWORD FROM REGISTRY
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
LOGIN INFO OBJECT
QUERY SERVER DATABASE RETRIEVE SECURITY TOKEN
RESULT MESSAGE OBJECT
SAVE SECURITY TOKEN
SERVER ACTIVATION OBJECT
RESULT MESSAGE OBJECT
UPDATE SERVER STATUS
CLOSE TCP CONNECTION

FIGURE 12

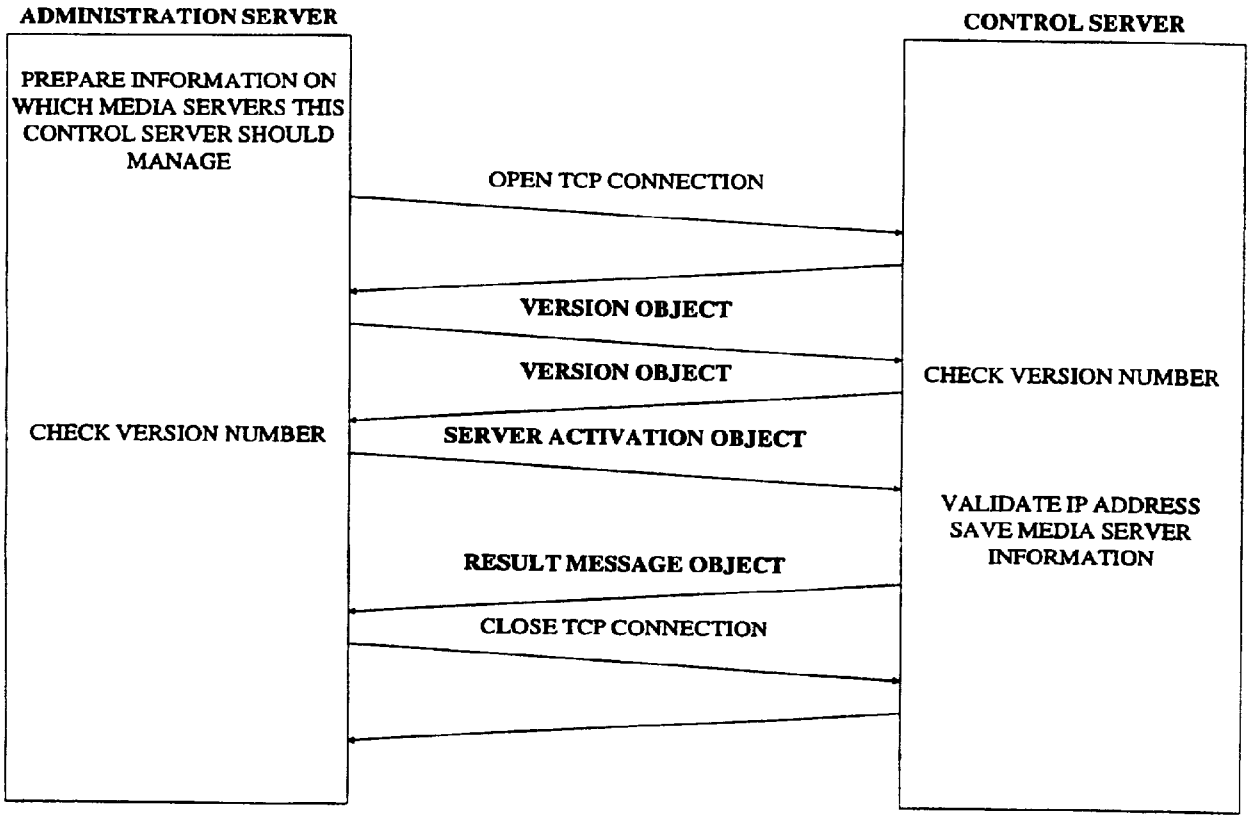

ADMINISTRATION SERVER
CONTROL SERVER
PREPARE INFORMATION ON WHICH MEDIA SERVERS THIS CONTROL SERVER SHOULD MANAGE
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
SERVER ACTIVATION OBJECT
VALIDATE IP ADDRESS SAVE MEDIA SERVER INFORMATION
RESULT MESSAGE OBJECT
CLOSE TCP CONNECTION

FIGURE 13

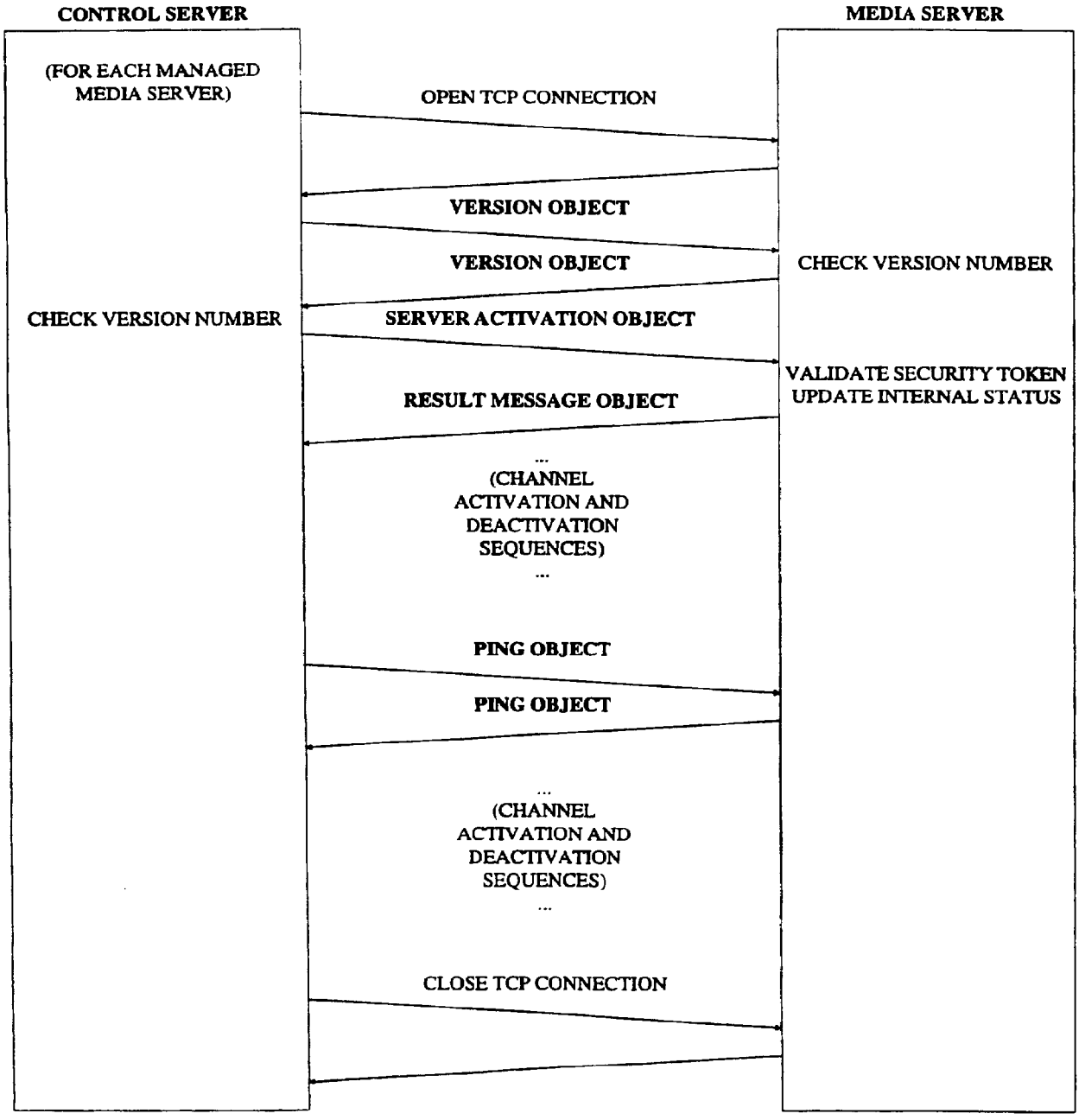

CONTROL SERVER
MEDIA SERVER
(FOR EACH MANAGED MEDIA SERVER)
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
SERVER ACTIVATION OBJECT
VALIDATE SECURITY TOKEN
UPDATE INTERNAL STATUS
RESULT MESSAGE OBJECT
...
(CHANNEL ACTIVATION AND DEACTIVATION SEQUENCES)
...
PING OBJECT
PING OBJECT
...
(CHANNEL ACTIVATION AND DEACTIVATION SEQUENCES)
...
CLOSE TCP CONNECTION

FIGURE 14

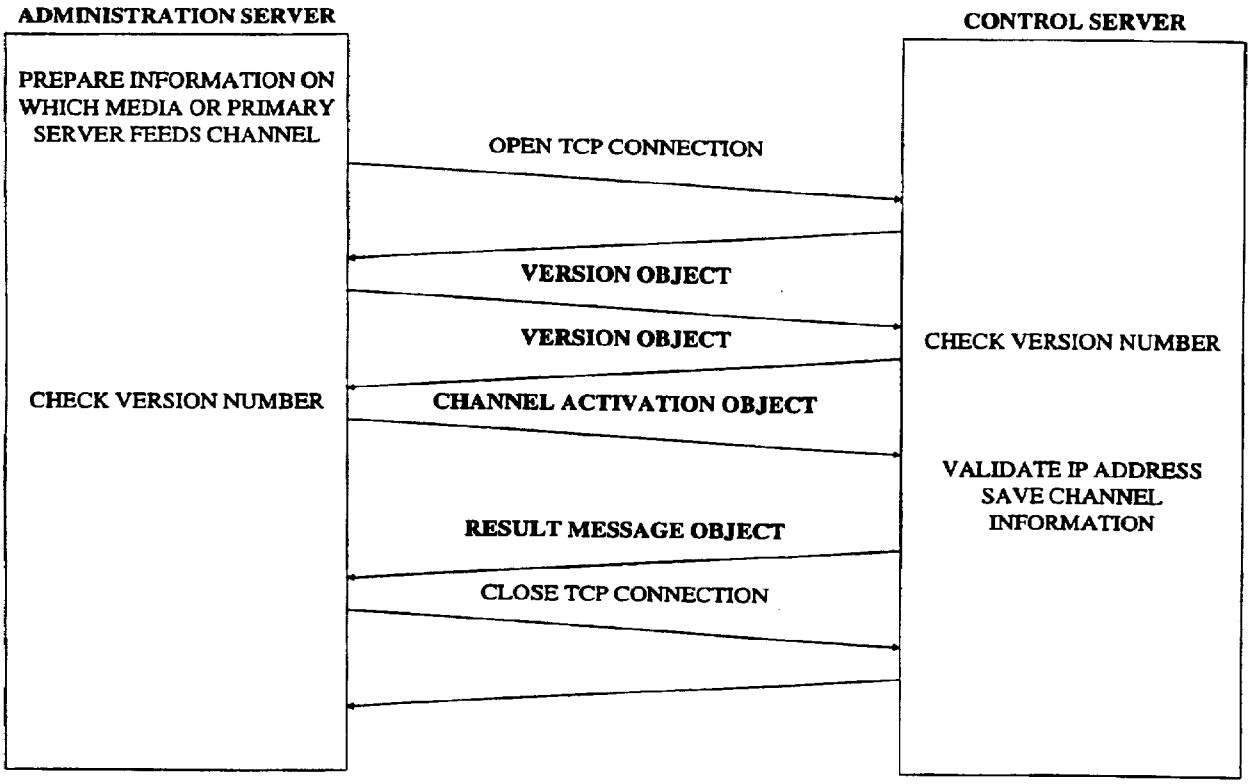

ADMINISTRATION SERVER
CONTROL SERVER
PREPARE INFORMATION ON WHICH MEDIA OR PRIMARY SERVER FEEDS CHANNEL
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
CHANNEL ACTIVATION OBJECT
VALIDATE IP ADDRESS SAVE CHANNEL INFORMATION
RESULT MESSAGE OBJECT
CLOSE TCP CONNECTION

FIGURE 15

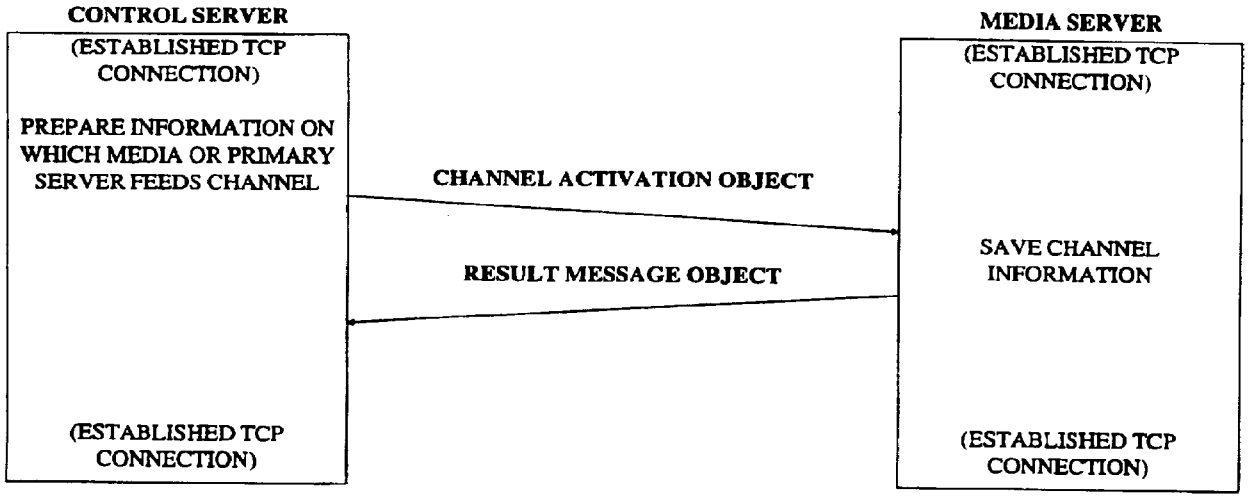

CONTROL SERVER
(ESTABLISHED TCP CONNECTION)
PREPARE INFORMATION ON WHICH MEDIA OR PRIMARY SERVER FEEDS CHANNEL
(ESTABLISHED TCP CONNECTION)
CHANNEL ACTIVATION OBJECT
RESULT MESSAGE OBJECT
MEDIA SERVER
(ESTABLISHED TCP CONNECTION)
SAVE CHANNEL INFORMATION
(ESTABLISHED TCP CONNECTION)

FIGURE 16A

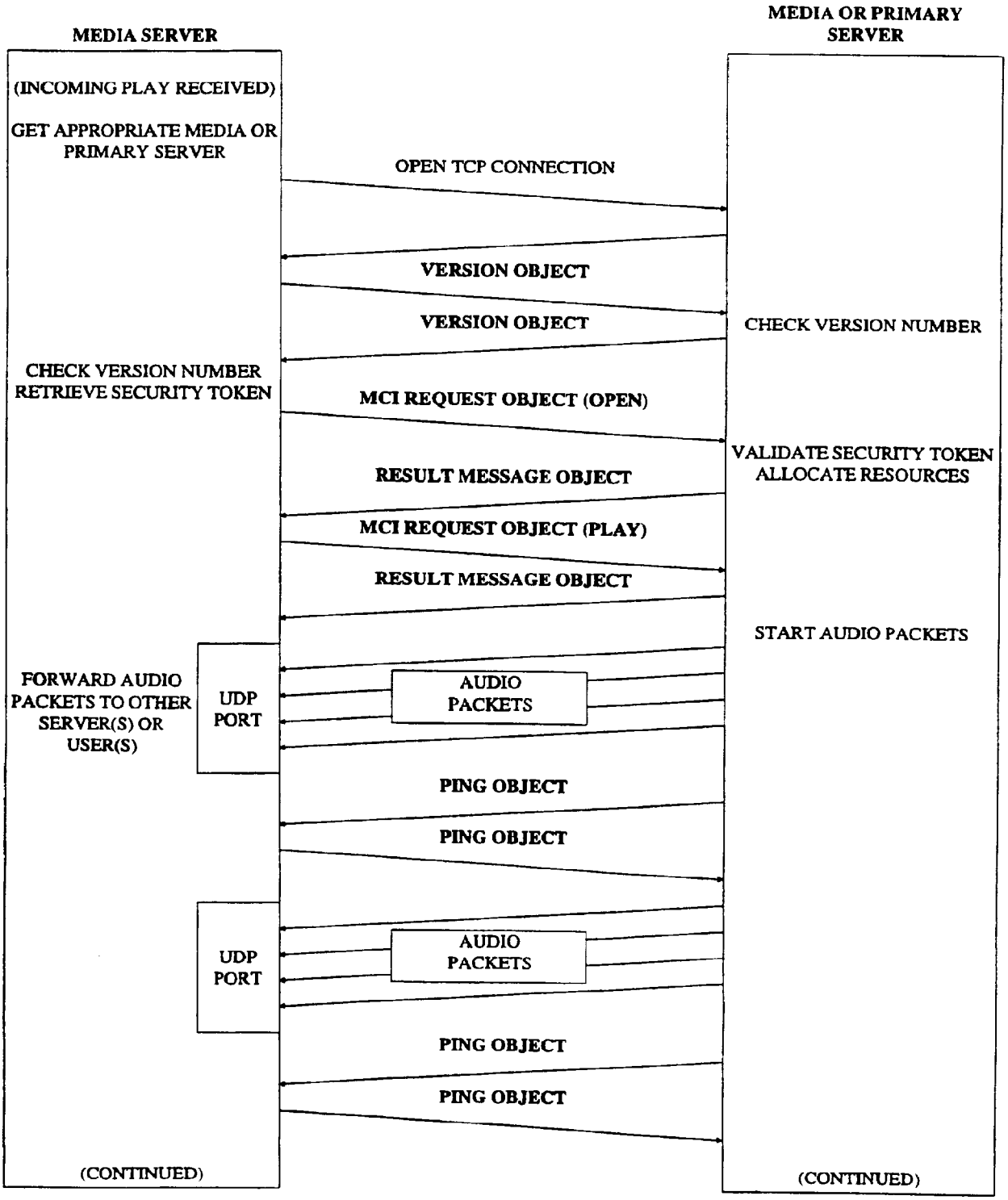

MEDIA SERVER
MEDIA OR PRIMARY SERVER
(INCOMING PLAY RECEIVED)
GET APPROPRIATE MEDIA OR PRIMARY SERVER
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER RETRIEVE SECURITY TOKEN
MCI REQUEST OBJECT (OPEN)
VALIDATE SECURITY TOKEN ALLOCATE RESOURCES
RESULT MESSAGE OBJECT
MCI REQUEST OBJECT (PLAY)
RESULT MESSAGE OBJECT
START AUDIO PACKETS
FORWARD AUDIO PACKETS TO OTHER SERVER(S) OR USER(S)
UDP PORT
AUDIO PACKETS
PING OBJECT
PING OBJECT
UDP PORT
AUDIO PACKETS
PING OBJECT
PING OBJECT
(CONTINUED)
(CONTINUED)

FIGURE 16B

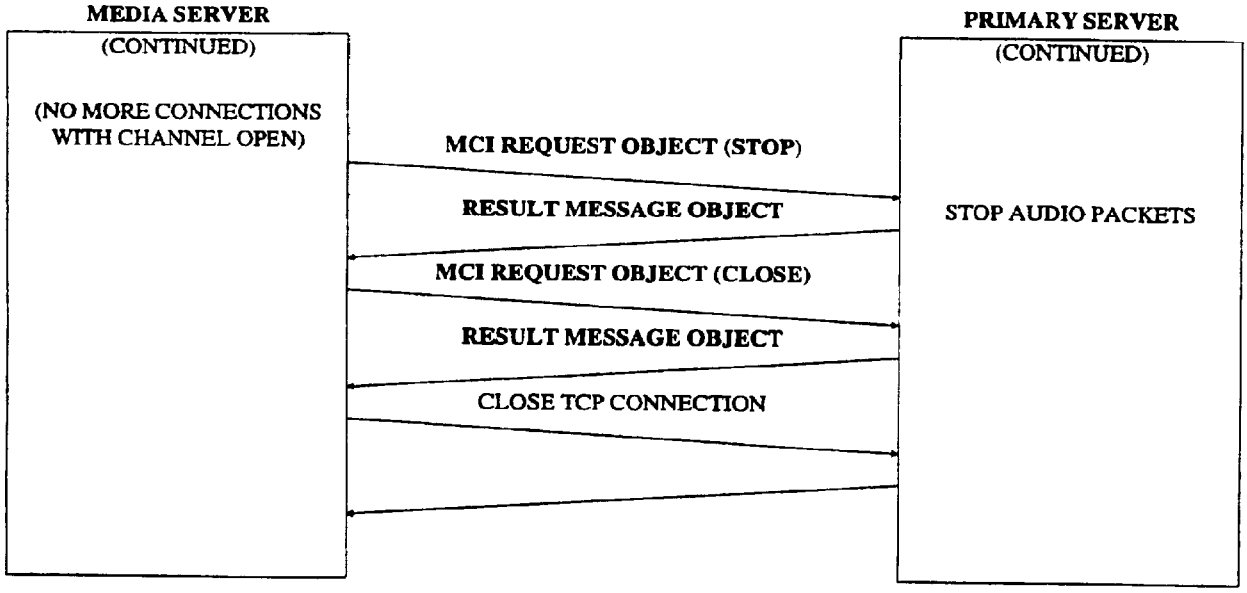

MEDIA SERVER
(CONTINUED)
(NO MORE CONNECTIONS WITH CHANNEL OPEN)
PRIMARY SERVER
(CONTINUED)
MCI REQUEST OBJECT (STOP)
RESULT MESSAGE OBJECT
STOP AUDIO PACKETS
MCI REQUEST OBJECT (CLOSE)
RESULT MESSAGE OBJECT
CLOSE TCP CONNECTION

FIGURE 17

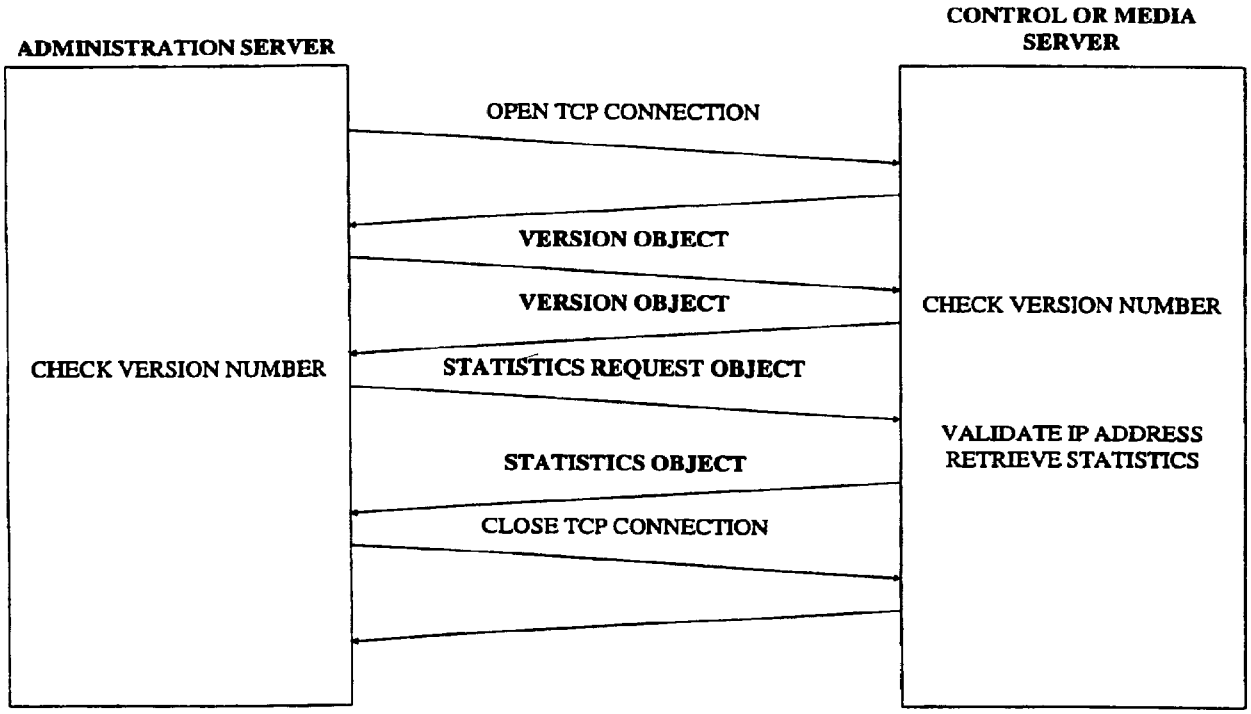

ADMINISTRATION SERVER
CONTROL OR MEDIA SERVER
OPEN TCP CONNECTION
VERSION OBJECT
VERSION OBJECT
CHECK VERSION NUMBER
CHECK VERSION NUMBER
STATISTICS REQUEST OBJECT
VALIDATE IP ADDRESS
RETRIEVE STATISTICS
STATISTICS OBJECT
CLOSE TCP CONNECTION

Figure 18
Main User Screen

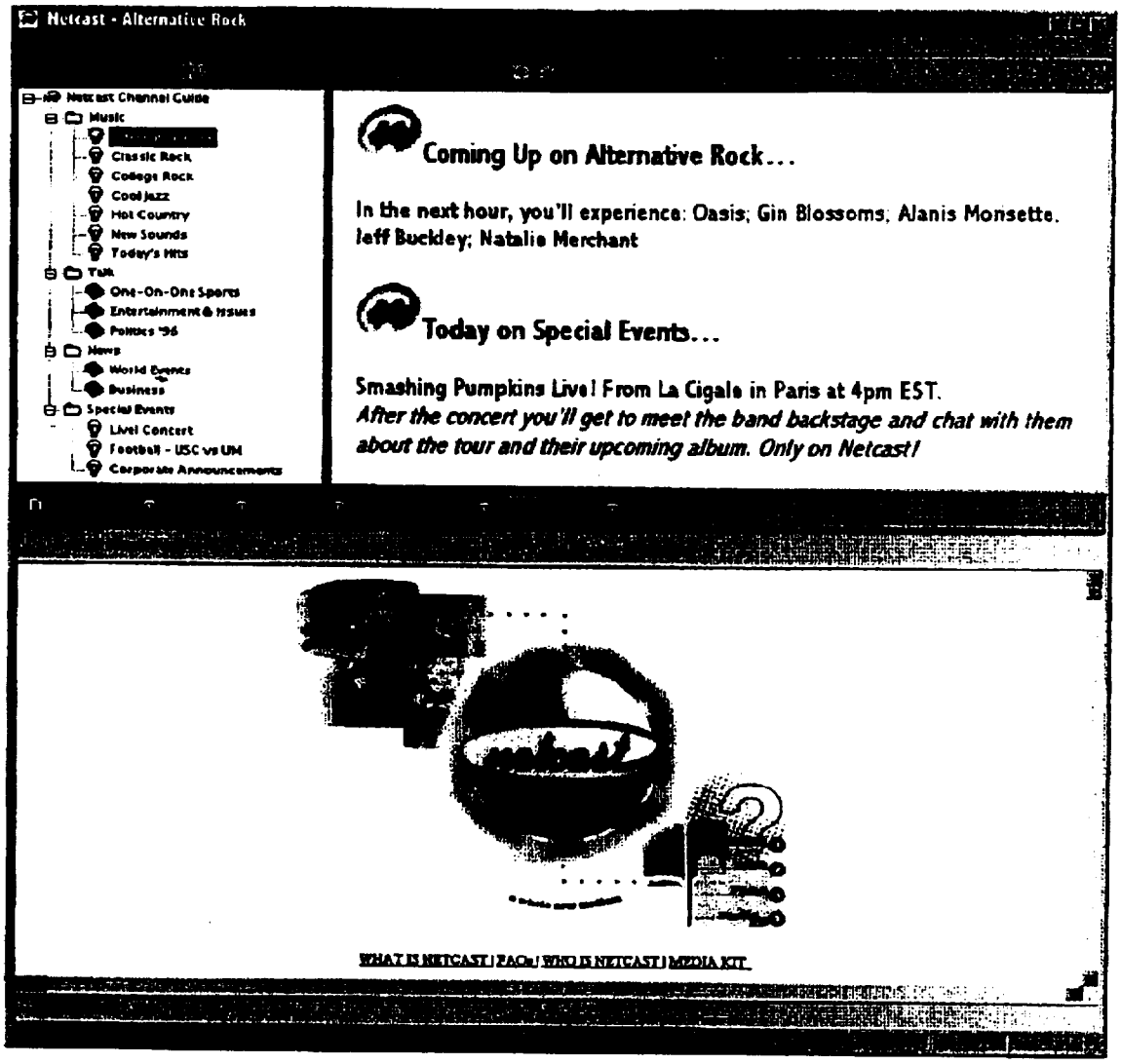
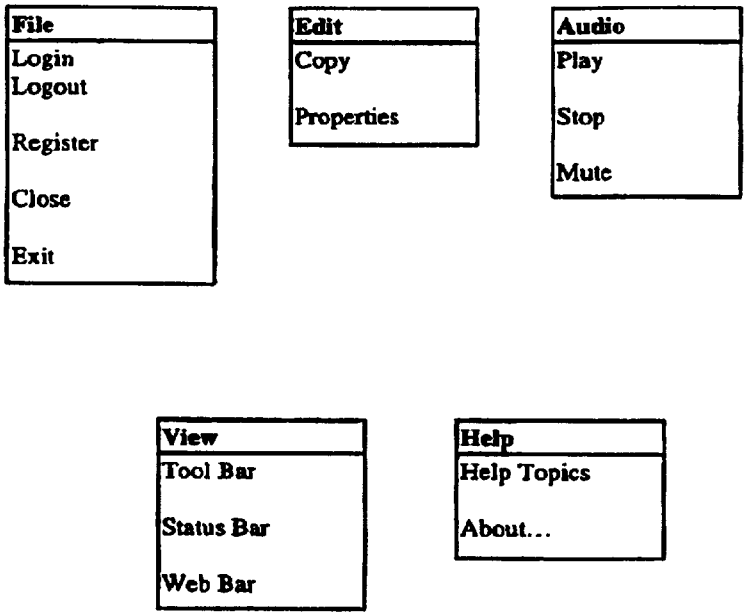


Figure 19
Key Pull-Down Menus on Main User Screen



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/07893

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/00
US CL :364/514A

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 364/514A; 348/13,561,110,565; 395/200.2, 200.09, 604

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,P	US, A, 5,617,565 (AUGENBRAUN et al.) 01 April 1997, columns 7-10.	1-37
Y,P	US, A, 5,604,542 (DEDRICK) 18 February 1997, columns 2-3.	1-37

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:	
"A" document defining the general state of the art which is not considered to be part of particular relevance	
"E" earlier document published on or after the international filing date	
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention	
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone	
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art	
"&" document member of the same patent family	

Date of the actual completion of the international search 19 JULY 1997	Date of mailing of the international search report 24 SEP 1997
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer ELLIS B. RAMIREZ Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet)(July 1992)*

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30, H04J 3/22	A1	(11) International Publication Number: WO 98/04985
		(43) International Publication Date: 5 February 1998 (05.02.98)

(21) International Application Number: PCT/US97/13097

(22) International Filing Date: 25 July 1997 (25.07.97)

(30) Priority Data:
60/022,598 25 July 1996 (25.07.96) US

(71)(72) Applicant and Inventor: KRIEGSMAN, Mark [US/US]; 581 Boylston Street #706, Boston, MA 02116 (US).

(74) Agent: GREELEY, Paul, D.; Ohlandt, Greeley, Ruggiero & Perle, Suite 903, One Landmark Square, Stamford, CT 06901 (US).

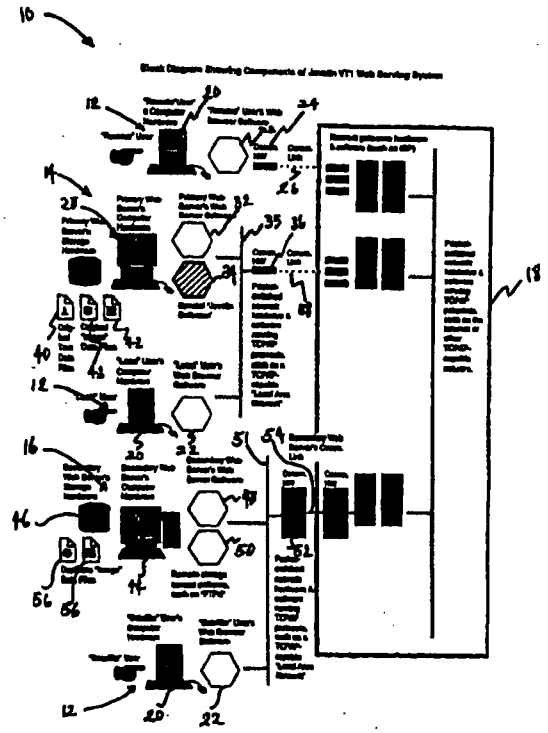
(81) Designated States: AL, AM, AT, AU, AZ, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GE, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, TJ, TM, TR, TT, UA, UG, UZ, VN, ARIPO patent (GH, KE, LS, MW, SD, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

Published
With international search report.

(54) Title: WEB SERVING SYSTEM WITH PRIMARY AND SECONDARY SERVERS

(57) Abstract

The present invention is a collaborative server system for providing high speed data transmission of data files across a communications network which comprises a communications network, a primary server (14) having a primary communications component (36) for connecting the primary server (14) to the network and at least one secondary server (16) having secondary communications component (52) for connecting the secondary server to the network. The primary and secondary servers (14, 16) include storage component for storing data files (40, 42). The data files include static data files (42) and/or dynamic data file (40). The storage component of primary server (14) stores at least one look-up table having specific criteria pertaining to the data files and the primary and secondary servers (14, 16). The processor of the primary server (14) is effective to receive a request for specific data files from a network user to look-up specific criteria in the look-up table.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

WEB SERVING SYSTEM WITH PRIMARY AND SECONDARY SERVERS

The present invention relates generally to a network of computer systems that transfer data files through a network connection, some of which are repeatedly transferred without changes ("static files"), and some of which may be modified for each transfer ("dynamic files"). More particularly, the present invention relates to web servers that are connected to a TCP/IP-capable network and transfer data files of the World Wide Web to computer systems connected to the network. In particular, the present invention is a web serving system that efficiently coordinates the efforts of two or more web servers to transfer data files rapidly through the TCP/IP-capable network while minimizing the cost of such system.

BACKGROUND OF THE INVENTION

A web serving system or web server is a computer system that runs web server software and connects to a communications network, such as the Internet. A publisher may use the web serving system to offer various data files for public access, including text, graphics, audio, video, and other types of data files. Thus, anyone having a computer that is connected to the same network may retrieve these data files offered by the publisher through the use of a standardized communications protocol. One protocol, namely the Hyper Text Transfer Protocol ("HTTP"), is commonly used on TCP/IP-capable networks to transfer data files of the World Wide Web.

Data files of the World Wide Web are transferred from a web server to a web browser computer via a TCP/IP-capable network. The web browser computer typically consists of a computer, web browser software running on the computer, and communications hardware to connect the computer to the network through a communications link. Likewise, the web server typically consists of a

computer, web server software running on the computer, storage hardware for storing the web pages, and communications hardware to connect the computer to the network through a communications link. A wide variety of communications links are available for connecting a computer to a TCP/IP-capable network, including an analog telephone connection, 56K line connection, ISDN connection, fractional T1 connection, Full T1 connection, cable-modem connection and satellite connection.

The user may view the contents of a specific data file of a particular web server by operating the web browser software. In particular, when a user wishes to receive the data file, the user operates the web browser computer so that it indicates the network address of the appropriate web server and the name of the data file on the server. The web browser computer then establishes a clear communications channel between the web server and itself, and requests the target data file. Next, the web server retrieves the specified data file from the web server's storage hardware and transmits the contents of the data file through the communications network to the web browser computer. The web browser computer then creates an audio/visual presentation of the data file for the user by activating the various hardware subsystems, such as one or more video monitors and/or audio speakers.

Web browser software has been enhanced to intermingle several pieces of text, graphics, video, audio, and animation together so that browsing or surfing the World Wide Web is an interactive and engaging process. In particular, standardized tags or sequences of text characters are included in the data files to control the placement of intermingled data files on the pages of the World Wide Web. A data file has text data and non-text data in which the standardized tags are included with the text data. The static data includes image data, animation data, video data, computer programs, and other types of non-text data. A data file containing a mixture of text and standardized tags is referred to as web page.

The standardized tags can include additional information, such as the web addresses, data file names of other web servers, and data files of graphic images, video clips, and audio recordings. Thus, most web browser software automatically presents an integrated display of dynamic data and embedded static data when displaying a webpage to the user. Accordingly, two distinct data files are retrieved to create a particular display: the first data file includes the text and tags, and the second data file includes the non-text data.

There are typically over a thousand web browsing users for every web server in operation and, thus, web servers are subjected to numerous requests for data files. Most contemporary web servers are capable of processing this volume of requests, but the web server's communications link quickly becomes saturated. In particular, any given communications link can only transmit a finite number of bits per second, and when a web server using all of the available bandwidth on its communications link to transmit data files, no more data files may be transmitted through that communications link until the transfers in progress are finished. Most communications links cannot sustain more than a dozen concurrent data file transfers, and each transfer can take several seconds.

Because of the high ratio of web browsers to web servers, it is extremely desirable to have a web server that is capable of handling at least ten concurrent data file transfers. Data communications technology, unfortunately, has not evolved as fast as other computer technologies. The least expensive type of communications link is the analog telephone line. However, the extremely constrained capacity of the analog telephone line makes it unacceptable for most web servers. The cost of other higher-speed communications links such as T1 lines increases along with their capacity, and they are far more expensive to setup and maintain than an analog or even ISDN telephone line.

On the other hand, because Full T1 lines are fairly cost effective, Internet Service Providers ("ISPs") have created profitable businesses by purchasing a Full T1 line, operating a web server, and renting out storage space to clients on their web server's storage hardware. These ISPs charge a low monthly rate for renting storage space on their web servers. Any data file that a client places in the rented storage space can be retrieved by web browsers that request them and, thus, clients make their pages and images accessible to a larger number of concurrent browsers without having to pay for the full cost of the full T1 themselves.

However, in order to maintain control and security of their web servers, the ISPs usually limit their clients to publishing static pages. Static pages are only stored and retrieved and, thus, dynamic and interactive pages are not permitted. Dynamic and interactive pages are essentially separate software programs that produce pages as their output. Since the software program is executed each time a web browser computer requests the page, the program may produce a different page each time, specifically for the web browser that made the request. This facility is becoming increasingly popular since it enables the interconnection of web server software and other software programs, such as databases, electronic commerce systems, e-mail systems, stock quote services, etc. Therefore, in order to have dynamic and interactive pages, a publisher cannot rent space from an ISP but must setup and maintain his or her own custom web server which is often prohibitively expensive.

Accordingly, the web serving system of the present invention has the capability of delivering text, graphics, video, audio and interactive multimedia over the Internet's World Wide Web at extremely high speeds, using only inexpensive, readily available hardware, software and services. The system requires no more effort than alternative systems, yet produces the highest performance-to-price ratio of any comparable World Wide Web publishing system.

SUMMARY OF THE INVENTION

The present invention is a collaborative server system capable of providing high speed data transmission of data files across a communications network which, in brief summary, comprises a communications network, a primary server having a primary communications means for connecting the primary server to the communications network, and at least one secondary server having a secondary communications means for connecting the secondary server to the communications network. The primary server and the at least one secondary server include means for storing data files and means for transmitting the data files to the communications network. The data files include static data files and/or dynamic data files. The storage means of the primary server further stores at least one look-up table having specific criteria pertaining to the data files and the primary and at least one secondary servers. The processor means of the primary server is effective to receive a request for specific data files from a network user, to look-up specific criteria in the look-up table pertaining to the specific data files, and to allocate transmission of each specific data file between the primary server and the at least one secondary server based on the specified criteria.

The storage means of the primary server includes static data files and dynamic data files, and the storage means of the secondary server includes a duplicate of the static data files. Thus, the processor means includes means for transferring the duplicate of the static data files from the storage means of the primary server to the storage means of the secondary server. Accordingly, the processor means of the primary server is capable of providing the dynamic files of the primary server and the duplicate of the static data files of either the primary server or the secondary server to the communications network in response to a single request for the static data files and dynamic data files of the primary server.

In addition, it is preferred that the secondary communications means of the secondary server transmits and receives communications signals to and from

certain user's computers at faster rates, with lower latency, more cost-effectively, or more efficiently than that of the primary communications means of the primary server. In particular, the primary communications means of the primary server is a lower-cost connection to the communications network (i.e., an analog communication line), and the secondary communication means is a connection with better bandwidth, latency, cost-effectiveness, and efficiency (i.e., a digital T1 communication line).

Further, the system includes at least one computer system that is connected to the communications network and has the capability of sending a request to the primary server. In addition, the processor means of the primary server includes means for determining an optimum server from the group of servers including the primary and secondary server to transmit the duplicate of the static data files to the computer system when two or more servers are available. The criteria for determining which one of the servers shall transmit includes transmission speed and available capacity of the primary communications means, proximity of the computer system to each server, availability of each server, version of the duplicate of the data file on each server and financial cost of transmitting data from each server.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and still further objects and advantages of the present invention will be more apparent from the following detailed explanation of the preferred embodiments of the invention in connection with the accompanying drawings:

Fig. 1 is a diagrammatic view of the preferred web serving system in accordance with the present invention;

Fig. 2. is a flow diagram representing the operation of the primary web server's computer when used with the supplemental web server software of Fig. 1;

Fig. 3 is a flow diagram of the DEMAND MO subroutine of Fig. 2;

Fig. 4 is a flow diagram of the BEST SI subroutine of Fig. 2;

Fig. 5 is a flow diagram of the REWRITE subroutine of Fig. 2;

Fig. 6 is a preferred flow diagram of the REWRITE subroutine of Fig. 2; and

Fig. 7 is a flow diagram of the POSTPROCESS subroutine of Fig. 2.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to the drawings, and in particular, to Fig. 1, there is provided a web serving system of the preferred embodiment which is generally represented by reference numeral 10. Web serving system 10 has the capability of delivering text, graphics, audio recordings, video recordings, computer animation, and computer programs and data to remote computer systems and storage devices through a distributed data network at extremely high speeds, often in excess of 300,000 bits per second, in a way that is interoperable with the World Wide Web's hypertext protocol. In particular, web serving system 10 combines a low-cost communications channel of a primary server with a more desirable communications channel of one or more secondary servers as well as computer storage and retrieval services for both the low speed and high speed channels. By coordinating data retrieval from the various servers, the primary server is able to control the data files sent by each server so that the speed in which data files are delivered to a target browser computer is maximized while retaining the ability to

provide dynamic and interactive pages (i.e., dynamic data files). In addition, the setup and maintenance costs of web serving system 10 are substantially less than that of conventional web serving systems since its secondary web server takes advantage of the low cost of renting or otherwise using storage space in existing servers and its primary web server takes advantage of lower-cost, lower-bandwidth communications links including analog telephone lines and lower-speed digital connections.

When used with an analog telephone line, preferred web serving system 10 shown in Fig. 1 is capable of fulfilling requests for over 100,000 data files per day at a minimal cost. The available capacity of the system is dependent on a number of factors and conditions, but the system can reliably sustain performance in the 400Kbps range and process many transfers concurrently. Also, the monthly cost per concurrent connection is much lower than that of a conventional web serving system that uses a 56K line connection, ISDN connection, fractional T1 connection, or Full T1 connection.

As shown in Fig. 1, preferred web serving system 10 includes one or more web browser computer systems 12, a primary web server 14 and one or more local web browser computer system 15 connected on a local area network (LAN) 35, one or more secondary web servers 16 and one or more satellite web browser computer system 15 connected on a LAN 51, and a communications network 18. Web browser computer system 12 may be a web user connected on a local area network (LAN) 35 with primary web server 14 (local user), connected on a LAN 51 with secondary web server 16 (satellite user) or not connected to either LAN 35 or LAN 51 (remote user).

Each web browser computer system 12 includes a user's computer 20, web browser software 22 residing in computer 20, communications hardware 24 connected to computer 20, and communications link 26 connecting

communications hardware 24 to communications network 18. In particular, user's computer 20 may be any type of computer system that is capable of running user's web browser software 22 and connecting to communications network 18. For example, user's computer 20 may be a personal computer or workstation that includes a memory portion for storing user's web browser software 22 and a processor for executing web browser software 22 so that the user may interact with software 22. Also, user's computer 20 is connected to user's communications hardware 24 which, in turn, is linked to communications network 18 by user's communications link 26. For example, a modem may be used to link user's computer 20 to an Internet Service Provider ("ISP") through a telephone line. Since a wide variety of different computer systems may connect to the network, it is to be understood that user's computer 20 of the present invention is not limited to the personal computer or workstation examples described above. Note that web browser computer systems 12 which are connected to either LAN 35 or LAN 51 respectively employ the same communications hardware and link as primary server system 14 and secondary server system 16.

Primary web server 14 includes a primary computer 28, storage hardware 30 connected to computer 28, web server software 32 and supplemental web server software 34 residing in computer 28, communications hardware 36 connected to computer 28, and communications link 38 providing a connection between communications hardware 36 and communications network 18. Also, primary storage hardware 30 may reside internal or external to primary computer 28 and is capable of storing primary data files, such as original dynamic data files 40 and original static data files 42. Dynamic data files 40 include standardized tags as well as text data, whereas static data files 42 typically include non-text data, such as image data, animation data, video data, audio data, and computer programs. Primary data files (40, 42) are commonly transmitted over packet-switched communications network 18 using TCP/IP protocols and read by user's web browser software 22. Primary computer 28 may be any type of computer

system that is capable of running primary web server software 32 and supplemental web server software 34, and connecting to communications network 18. A wide variety of computer systems, ranging from mainframe computer to personal computers, are capable of running web server software 32 with supplemental web server software 34 and, thus, primary computer 28 is affordable and readily available. For example, primary computer 28 may be a personal computer or workstation that includes a memory portion for storing primary web server software 32 and supplemental web server software 34, and a processor for executing both software. Also, similar to web browser computer system 12, primary computer 28 is connected to primary communications hardware 36 which, in turn, is linked to communications network 18 by primary communications link 38.

In addition, the present invention includes one or more secondary web servers 16. Although a secondary web server 16 may be purchased and setup for the present invention, the services of such secondary web server 16 is readily available to avoid their high setup and maintenance costs, which is a feature of the present invention. For example, one may rent space on a secondary web server from an Internet Service Provider for a low fee, or arrange to use space on a secondary web server owned by an organization or university.

Typically, each secondary web server 16 includes a secondary computer 44, storage hardware 46 connected external or provided internal to secondary computer 44, web server software 48 and remote storage access software 50 residing in secondary computer 44, communications hardware 52 connected to secondary computer 44, and communications link 54 connecting secondary computer 44 to communications network 18. Secondary storage hardware 46 is capable of storing data files 56 that are duplicates of the original data files (40, 42) in primary storage hardware 30 of primary web server 14. Also, secondary communications link 54 has superior operating characteristics than that of primary

communications link 38. For example, as shown in Fig. 1, secondary communications link 54 is a Full T1 line that has higher performance capacity than primary communications link 38 which is an analog telephone line. Also, similar to primary computer 28, secondary computer 44 may be any type of computer system, ranging from personal computer to mainframe computers, that is capable of running secondary web server software 48 and remote storage access software 50, and connecting to communications network 18. Further secondary computer 44 is connected to secondary communications hardware 52 which, in turn, is linked to communications network 18 by secondary communications link 54.

Computer 28 of the primary web server 14 executes supplemental web server software 34 to perform the critical function of coordinating file transfers by primary web server 14 and secondary web server or servers 16. Computer 28 and supplemental web server software 34 exploit the fact that the ultimate audience, namely the users who accessing primary web server 14, for data files (40, 42) is not involved in the retrieval of each data file individually. In actuality, the task of retrieving data files, particularly static data files 42 that are embedded in pages, is performed automatically by the user's web browser software, such as web browser software 22.

Primary computer 28, when used with supplemental web server software 34, adds significant and powerful capabilities to primary web server software 32. In particular, primary computer 28 identifies the static files (i.e., typically non-text data files) that are consuming most of the capacity of primary communications link 38, copies those capacity-consuming static data files 42 to one or more secondary web servers 16 (which have higher capacity communications links), and then dynamically re-writes requested pages so that the embedded static data files are retrieved directly from the best or optimal web server each time. Primary computer 28 also records every request for a data file (40, 42), builds a statistical

model of how the capacity of primary communications link 38 is being consumed by the transmission of the various data files (40, 42), determines which static data files 42 are consuming the most capacity, uses standardized communication protocols to communicate through the channel between primary and secondary web servers (14, 16), uses the communication channel to copy static data files 42 from primary storage hardware 30 to secondary storage hardware 46, and tracks the progress and location of all duplicated files 56 through the use of a database.

In addition, primary computer 28 responds to requests for pages stored on the primary storage hardware 30 by reading the data file (40, 42) from the primary storage hardware 30, finding all tags in the data file (40, 42) that refer to static data files 42 stored on primary web server 14, examining the database to retrieve a list of secondary web servers 16 and data files 56 where instantiation of static data file 56 referred to by each tag may be found. Primary computer 28 also responds to requests for pages stored on primary storage hardware 30 by determining, for each static data file 42 referred to by the page's tags, which available instantiation would be the best instantiation to transmit to web browser computer system 12. Considerations for determining the best instantiation includes the transmission speed of each web server's communications link (38, 54), the currently available capacity of each web server's communications link (38, 54), the proximity of web browser computer system 12 to each web server (14, 16) given the topology of the communications network 18, the current availability or operability of each secondary web server 16, the version of the data file 42 available on each web server (14, 16) in which only instantiations that are current are eligible, the cost of transmitting the data file from each server, the response latency of each server with respect to the requesting computer and the efficiency of transmitting the data file from each server. Primary computer 28 further responds to requests for pages stored on the primary storage hardware 30 by creating a re-written page, in which each tag that refers to a particular data file 42 is replaced with a tag that is functionally identical except that the tag now refers to the best instantiation rather

than to original data file 42. Also, the re-written page is transmitted through the primary connections link to web browser computer system 12, statistical information about the transmission of the page and the supposed transmission of any static data file 42 whose best instantiation is not on primary web server 14 is recorded, various statistics are used to update the database with revised information about the usage levels and patterns of data files (40, 42, 56) on both primary web server 14 and secondary web servers 16, and statistical information in the database is used to guide primary computer 28 in optimizing of the distribution of data file instantiations across multiple servers.

Once static data files 42 have been copied to one or more secondary web servers 16, only original dynamic data files 40, particularly their text-and-tags information, need to be transferred through primary communications link 38. Most static data files 56 that are requested from outside the local area network are transmitted over the higher-speed, higher-capacity, lower-latency or more efficient links of secondary communications hardware 52. If one secondary web server 16, whose communications link 54 is a Full T1 line, is used, the speed at which the primary computer 28 can deliver pages and static data files can be as high as about 1200Kbps for short bursts and tends to run consistently about 300Kbps to about 400Kbps. Thus, by combining the resources of a low-cost, low-bandwidth communications link 38 and some low-cost storage space on one or more secondary web servers 16 with high speed, high capacity communications links 54, the result is a web serving system which is fast, interactive and inexpensive.

It is important to note that the present invention provides high capacity performance, reliably sustaining about 400 Kilobits per second of capacity, while providing the capability of handling dynamic and interactive pages and benefiting from the economical startup and maintenance costs of a low cost, low bandwidth communications link. In particular, when used with an analog telephone line, the present invention is capable of delivering performance that is comparable to a

system having a fractional T1 connection and surpasses a system having 56K line or ISDN connection. Therefore, for the present invention, an expensive digital communications link, such as fractional T1 connection or Full T1 connection, is not required for high capacity performance. In fact, the primary communications link may be an analog telephone line connected to the primary communications hardware, such as a modem, and yet provide performance that is comparable to a fractional T1 connection and outperform a 56K line or ISDN connection.

Referring to Fig. 2, there is provided the main flow chart, beginning with step 60, for the process performed by primary computer 28, when using supplemental web server software 34, for coordinating the file transfer functions of primary web server 14 and secondary web server or servers 16. When a new request for a data file (40, 42) is received by web server software 32 as shown in step 62, primary computer 28 starts by examining the name of the requested file as using subroutine DEMAND MO shown in step 64. The coordinating means then selects the instance with the highest quality that is associated with this master object by using subroutine BEST SI as shown in step 66.

If the best instance is not associated with the primary server, then a redirect response indicating the location of the instance master object is prepared as shown in steps 68 and 70. If the best instance is associated with the primary server but the requested data file is not a page that may contain tags or reference to other data files, a direct response including the contents of the instance master object is prepared as shown in steps 72 and 74. If the best instance is associated with the primary server and the requested data file contains tags or reference to other data files, then a revised version of the contents of the instance master object is generated using subroutine REWRITE as shown in step 76. Then, as shown in step 78, a direct response including the rewritten contents of the instance master object is prepared. Next, in all cases, the contents of the instance found on the primary server are transmitted to the web browser that requested it as shown in

step 80. Finally, statistical information is recorded using subroutine POSTPROCESS and primary computer 28 stops execution as shown in steps 82 and 84.

Referring to Fig. 3, the subroutine DEMAND-MO (step 64 of Fig. 2) begins at step 86 and responds to the request for the data file (40, 42) received by web server software 32 as shown in step 88. Primary computer 28 then searches the database for a master object record that matches the name given as shown in step 90. If there is a matching master object record in the database, then the coordinating means returns the master object record as shown in steps 92 and 94. If a matching record is not in the database, a new master object record with the name of the given file is created in the database as shown in step 96. Next, a new statistics record is created for this master object, and an instance record with the name of the given file is created in the instance database as shown in step 98. Also, the instance record is marked as primary and local and the instance is associated with primary web server 14. Finally, as shown in step 94, the new master object record is returned and, as shown in step 100, subroutine DEMAND MO returns to the main process of Fig. 2.

Referring to Fig. 4, there is provided the BEST SI subroutine shown as step 66 in Fig. 2, starting with step 102. Primary computer 28 initially receives a given master object and retrieves the list of instances associated with the given master object from the database as shown in steps 104 and 106. As shown in steps 108, primary computer 28 determines whether there are any more matching records. If no more matching records exist then the primary computer 28 returns the instance master object having the lowest serving time and/or highest quality score, and the BEST SI subroutine returns to the main process of Fig. 2 as shown in steps 110 and 120.

If there are any more matching records, such record is retrieved as shown in step 112. Then, the serving time is calculated by considering the transmission time of the data file from the given instance, given the current load on the server for the instance and the load and maximum speed of the communications link (38, 54) for that server (14, 16). The quality score is set to be an inverse function of the time required, and scaled by the topological distance between the web browser computer system 12 making the request and the server (14, 16) for this instance. Cost, latency and efficiency are also considered part of the quality score. If the serving time is faster than any other serving time considered so far for this master object, then the instance is designated as having the lowest serving time, and primary computer 28 searches for the next record matching the master object in the database as shown in steps 116 and 118. Otherwise, if the serving time is not the fastest one considered, then primary computer 28 simply goes back to searching for the next record matching the master object in the database as shown in steps 116.

Referring to Fig. 5, there is provided the REWRITE subroutine shown as step 76 in Fig. 2, starting with step 200. Initially, the instance record is received, and a working copy of the data file of the instance record location is retrieved, as shown in steps 202 and 204. Primary computer 28 then analyzes the working copy of the data file for the next embedded reference to a data file as shown in step 206. Primary computer 28 determines whether an embedded reference was found in step 208. If not, the working copy is returned without any modifications as shown in step 210. If an embedded reference is discovered, the DEMAND MO subroutine is used to find the master object record for the embedded data file as shown in step 212. Next, the BEST SI subroutine is used to find the best instance for that master object as shown in step 214. Then, in the working copy of the data file, the reference to the original Web Server (i.e., an address code) and Data File names are replaced with the names of the web server and data file where the best instance is located a shown in step 216. If there is more in the data file to read,

then primary computer 28 will try to read the next embedded reference to a data file. Finally, the primary computer 28 transmits the contents of working file to the web browser that requested it as shown in step 210.

Referring to Fig. 6, there is provided a preferred embodiment of the REWRITE subroutine shown as step 76 in Fig. 2, starting with step 122. Initially, the instance record is received, and the output file is initialized so that it is empty as shown in steps 124 and 126. The next segment of text is then read from the data file as shown in step 128. Then in step 130, primary computer 28 determines whether any data was read from the file. If not, the contents of the output file are returned as shown in step 132. However, if data was read from the file, then all non-tag text is copied to the page output buffer a shown in step 134. When a tag is encountered, primary computer 28 tries to read the next tag-only segment of the incident master object from storage as shown in step 136. Next, as shown in step 138, the primary computer 28 determines whether data was read from the file. If not, then the contents of the output file are returned as shown in step 132. As shown in steps 138 and 140, if data was read from the file, the primary computer 28 determines whether there is a reference to another data file. If none is found, the tag is copied to the output file without modification as shown in step 142. However, if the tag could contain such references, the DEMAND MO subroutine is used to find the master object record for the name found as shown in step 144. Next, the BEST SI subroutine is used to find the best instance for that master object as shown in step 146. Then, in the tag, the reference to the original Web Server and Data File names are replaced with the names of the web server and data file where the best instance is located a shown in step 148. The tag is then appended to the page output buffer as shown in step 150. If there is more in the data file to read, then primary computer 28 will try to read the next non-tag segment. Finally, the primary computer 28 transmits the contents of the page output buffer to the web browser that requested it as shown in step 132.

Referring to Fig. 7, there is provided the POSTPROCESS subroutine shown as step 82 in Fig. 2, starting with step 154. Primary computer 28 updates the statistical database records for the instance, the master object, the server, and the communications links appropriate to the instance that was selected as shown in step 156. Then, the first replication server is obtained as shown in step 158, and a determination is made as to whether there is a server to consider as shown in step 160. If not, then subroutine POSTPROCESS returns to the main process of Fig. 1. If a server to consider does exist, then a determination is made of whether there is already an instance of this master object on this server as shown in step 162. If so, the next replication server is considered as shown in step 158. If there is not an instance of the master object on the server, primary computer 28 determines whether the master object is larger than the total storage capacity of this server as shown in step 164. If so, the next replication server is considered a shown in step 158.

If the master object is not larger than the total storage capacity of the server, a list is made of the instances, if any, that would have to be deleted from this server in order to accommodate the master object under consideration as shown in step 166. The total replication value of the list and the replication value of the master object are computed as shown in steps 168 and 170. The total replication value of the list is compared to the replication value of the master object as shown in step 172. If the replication value of the given master object is not greater than the total replication value of the list, then the next replication server is examined as shown in step 158. If the replication value of the given master object is greater than the total replication value of the list, then the remote instances on the list are deleted. Specifically, elements of the list are deleted from the given server and from the instance database as shown in step 174, and a copy of the master object is made onto the given server and the record of instance master object is added to the instance database as shown in step 176. If all secondary webservers 16 are to be treated as one pooled storage area, then the

POSTPROCESS subroutine returns to the main process of Fig. 2 as shown in step 178. If not, then more servers are examined as shown in step 158.

The present invention having been described with particular reference to the preferred forms thereof, it will be obvious that various changes and modifications may be made therein without departing from the spirit and scope of the invention as defined in the appended claims.

20

WHAT IS CLAIMED IS:

1. An apparatus connected to a network, for receiving a request for data files from a network user and allocating transmission of the data files between predetermined devices connected to the network, said apparatus comprising:

memory means for storing data files and at least one look-up table having specific criteria pertaining to said data files and said predetermined devices, said data files including dynamic data files and/or static data files; and

data allocation means which is capable of:

(i) receiving a request for specific data files, from a network user,

(ii) looking up specific criteria in said look-up table pertaining to said specific data files, and

(iii) allocating transmission of each specific data file between said predetermined devices based on said specific criteria; and

means for transmitting data to said network user.

2. The apparatus as recited in claim 1, wherein said data allocation means causes said transmitting means to transmit any dynamic data file in said specific data files to said network user.

3. The apparatus as recited in claim 1, wherein said data allocation means selects an optimum device from said determined devices for each static data file in said specific data file and causes said network user to request said each static data file from a corresponding selected optimum device.

4. The apparatus as recited in claim 3, wherein said data allocation means transmits an address code of the corresponding selected optimum device for each static data file to said network user.

5. The apparatus as recited in claim 4, wherein said network user, responsive to said receipt of said address code for each static data file, requests each of said static data files from the corresponding selected optimum device

6. The apparatus as recited in claim 1, further comprising means for transferring a copy of any data file stored in said memory means to any of said predetermined devices.

7. The apparatus as recited in claim 1, further comprising means for updating said specific criteria in said look-up table of said memory means.

8. The apparatus as recited in claim 1, wherein said specific criteria include a latency of a connection between each of said predetermined devices and said network user.

9. The apparatus as recited in claim 1, wherein said specific criteria include an estimated cost of transmitting the data file from each of said predetermined devices to the network user.

10. The apparatus as recited in claim 1, wherein said specific criteria include a transmission bandwidth of each of said predetermined devices.

11. The apparatus as recited in claim 1, wherein said specific criteria include a current capacity of each of said predetermined devices to transmit data files to said network user.

12. The apparatus as recited in claim 1, wherein said specific criteria include a version type of data files retrievable by each of said predetermined devices.

13. The apparatus as recited in claim 1, wherein at least one of said specific data files includes embedded references to additional data files.

14. The apparatus as recited in claim 1, wherein said data allocation means is further capable of looking up specific criteria in said look-up table pertaining to said additional data files, and allocating transmission of each of said additional data files between said predetermined devices, based on said specific criteria pertaining to said additional data files.

15. The server system as recited in claim 14, wherein said embedded references include corresponding address codes for each of said additional data files, said allocation means having means for rewriting said address codes.

16. A collaborative server system for providing high speed data transmission of data files across a network, comprising:

a primary server and a secondary server, both connected to a network across a communication medium, said primary server comprising: (a) memory means for storing data files and at least one look-up table having specific criteria pertaining to said data files and said primary server and said at least one secondary server, said data files including dynamic data files and/or static data files; (b) data allocation means which is capable of: (i) receiving a request for specific data files from a network user, (ii) looking up specific criteria in said look-up table pertaining to said specific data files, and (iii) allocating transmission of each specific data file between said primary server and said at least one secondary server based on said specific criteria; and (c) means for transmitting data across said network,

said at least one secondary server comprising: secondary memory means for storing data files, said data files including dynamic data files and/or static data files, and means for transmitting data across said network.

17. The server system as recited in claim 16, wherein said data allocation means causes said transmitting means of said primary server to transmit any dynamic data file in said specific data files to said network user.

18. The server system as recited in claim 16, wherein said data allocation means selects an optimum server from a group including said primary server and said secondary server for each static data file in said specific data file and causes said network user to request said each static data file from a corresponding selected optimum device.

19. The server system as recited in claim 18, wherein said data allocation means transmits an address code of the corresponding selected optimum device for each static data file to said network user.

20. The server system as recited in claim 19, wherein said network user, responsive to said receipt of said address code for each static data file, requests each of said static data files from the corresponding selected optimum device

21. The server system as recited in claim 16, wherein said primary server further comprises means for transferring a copy of any data file stored in said primary memory means to said secondary memory means of said at least one secondary server.

22. The server system recited in claim 16, wherein said primary server further comprises means for updating said specific criteria in said look-up table of said primary memory means.

23. The server system as recited in claim 16, wherein said specific criteria include a latency of a connection between said primary server and said at least one secondary server and said network user.

24. The server system as recited in claim 16, wherein said specific criteria include an estimated cost of transmitting the data file from said primary server and said at least one secondary server to the network user.

25. The server system as recited in claim 16, wherein said specific criteria include a transmission bandwidth of said primary server and said at least one secondary server.

26. The server system as recited in claim 16, wherein said specific criteria include a current capacity of said primary server and said at least one secondary server to transmit data files to said network user.

27. The server system as recited in claim 16, wherein said specific criteria include a version type of data files retrievable by said primary server and said at least one secondary server.

28. The server system as recited in claim 16, wherein at least one of said specific data files includes embedded references to additional data files.

29. The server system as recited in claim 28, wherein said data allocation means is further capable of: looking up specific criteria in said look-up table pertaining to said additional data files, and allocating transmission of each of said additional data files between said predetermined devices, based on said specific criteria pertaining to said additional data files.

30. The server system as recited in claim 29, wherein said embedded references include corresponding address codes for each of said additional data files, said data allocation means having means for rewriting said address codes.

31. The server system as recited in claim 16, wherein said primary server is connected to the network across a slower transmission medium than said at least one secondary server.

32. The server system as recited in claim 31, wherein said primary server is connected to the network across an analog transmission line and said at least one secondary server is connected to the network across a faster digital transmission line.

33. The server system as recited in claim 16, wherein said network user includes means for requesting and receiving data files, via the network.

34. The server system as recited in claim 33, wherein said network user further includes means for generating a web page from said specific data files.

35. The server system as recited in claim 16, wherein said network user and said primary server are connected in a local area network (LAN), said LAN being connected to said network.

36. The server system as recited in claim 16, wherein said network user and said at least one secondary server are connect to a local are network (LAN), said LAN being connected to said.

37. An apparatus connected to a network, for receiving a request for data file having embedded therein references to additional data files from a network user and allocating transmission of all of the data files between predetermined devices connected to the network, said apparatus comprising:

memory means for storing data files and at least one look-up table having specific criteria pertaining to said data files and said predetermined devices, said data files including dynamic data files and/or static data files; and

data allocation means which is capable of:

(i) receiving a request for at least one specific data file having embedded therein references to additional data files, from a network user,

(ii) looking up specific criteria in said look-up table pertaining to each of said additional files, and

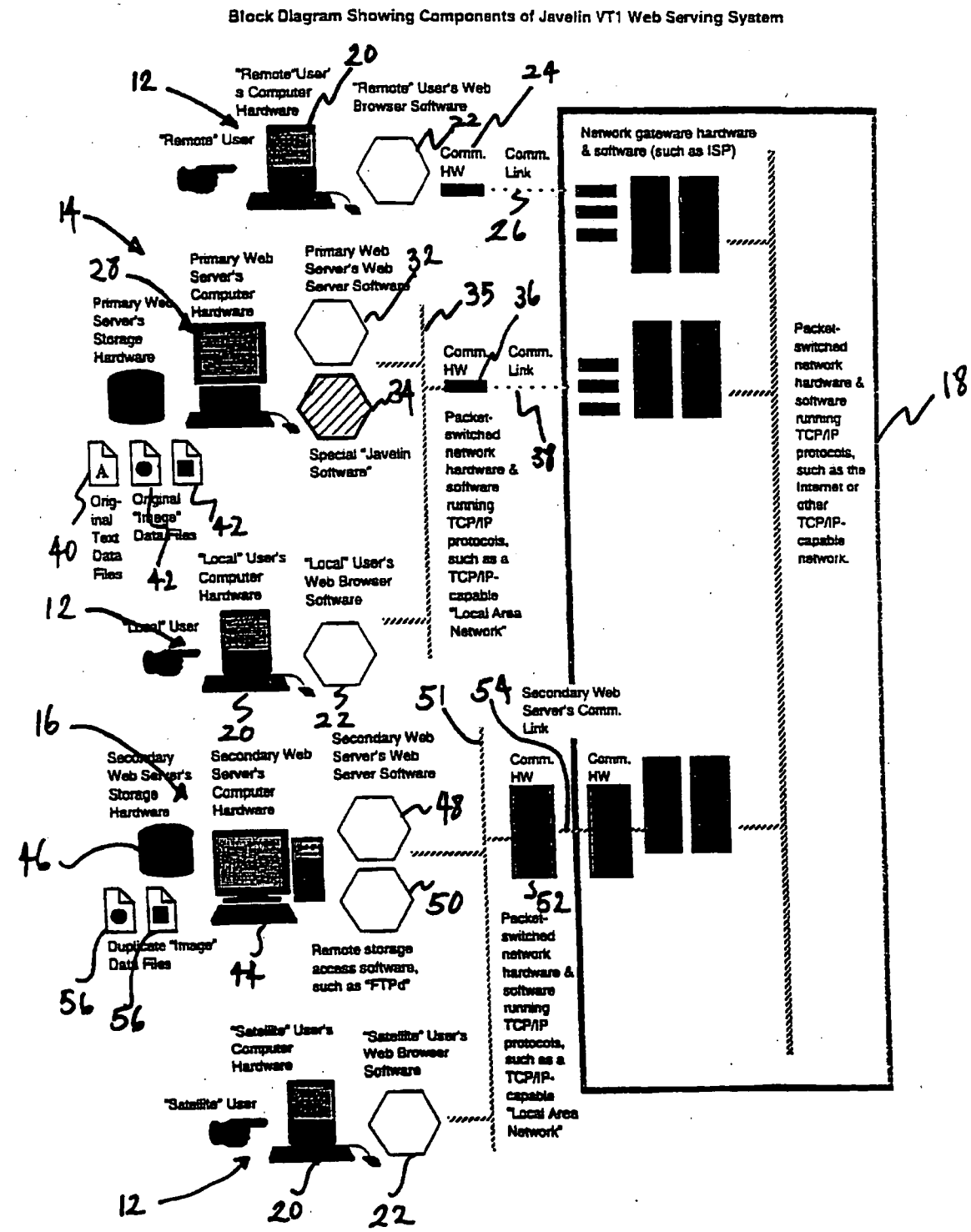
(iii) allocating transmission of each of said at least one specific data file and each of said additional data files between said predetermined devices based on said specificcriteria; and

means for transmitting data to said network user.

38. The apparatus as recited in claim 37, wherein said embedded references include corresponding address codes for each of said additional data files, said data allocation means having means for rewriting said address codes.

1/7

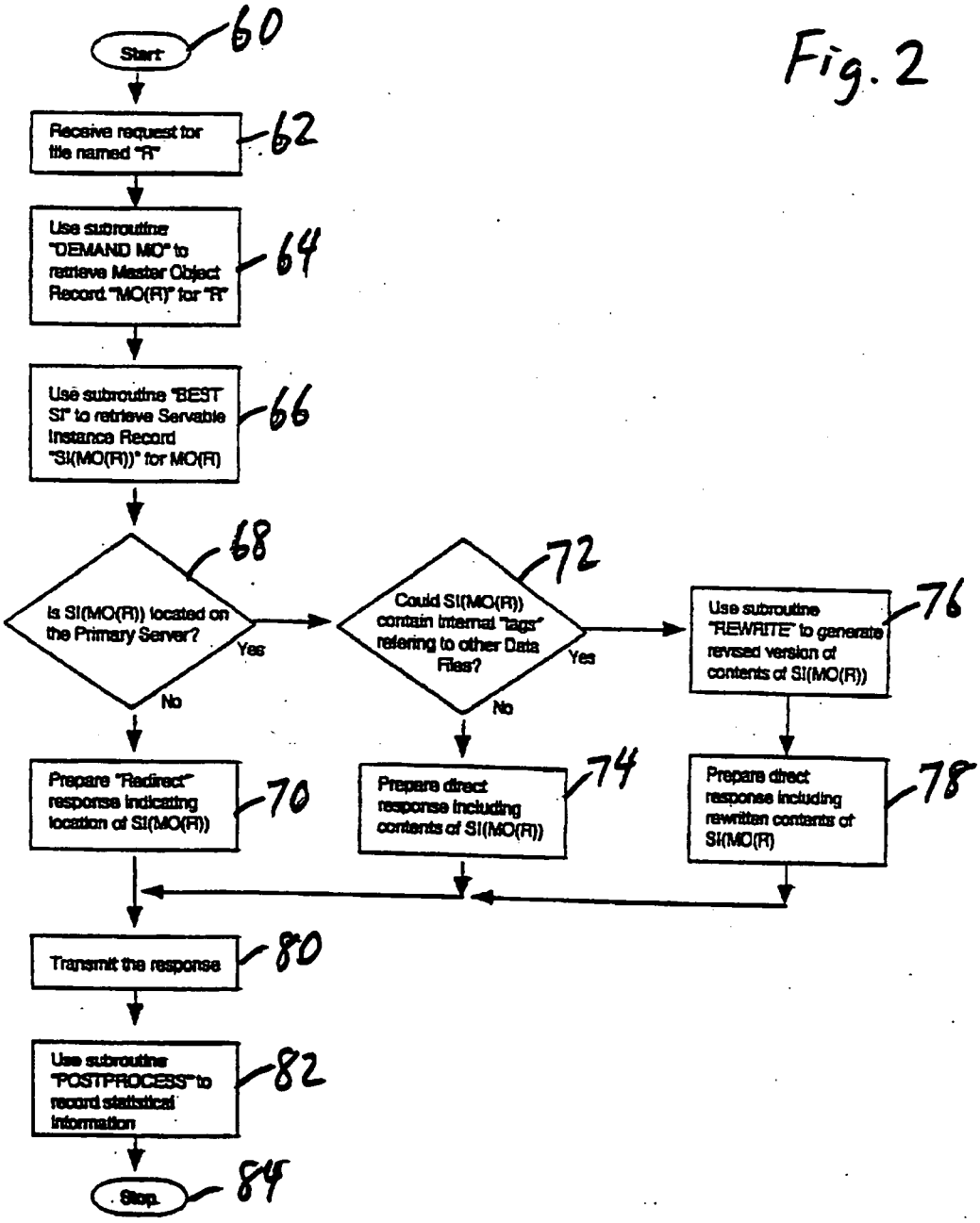
FIG. 1


Block Diagram Showing Components of Javelin VT1 Web Serving System
10
12
"Remote" User
20
"Remote" User's Computer Hardware
"Remote" User's Web Browser Software
24
22
Comm. HW
Comm. Link
26
Network gateware hardware & software (such as ISP)
14
28
Primary Web Server's Storage Hardware
Primary Web Server's Computer Hardware
Primary Web Server's Web Server Software
32
35
36
Comm. HW
Comm. Link
34
Special "Javelin Software"
Packet-switched network hardware & software running TCP/IP protocols, such as a TCP/IP-capable "Local Area Network"
38
Packet-switched network hardware & software running TCP/IP protocols, such as the Internet or other TCP/IP-capable network.
18
Original Text Data Files
Original "Image" Data Files
40
42
42
"Local" User's Computer Hardware
"Local" User's Web Browser Software
12
"Local" User
20
22
16
Secondary Web Server's Storage Hardware
Secondary Web Server's Computer Hardware
Secondary Web Server's Web Server Software
51
54
Secondary Web Server's Comm. Link
Comm. HW
Comm. HW
48
46
50
52
Duplicate "Image" Data Files
Remote storage access software, such as "FTPd"
Packet-switched network hardware & software running TCP/IP protocols, such as a TCP/IP-capable "Local Area Network"
56
56
44
"Satellite" User's Computer Hardware
"Satellite" User's Web Browser Software
"Satellite" User
12
20
22

2/7

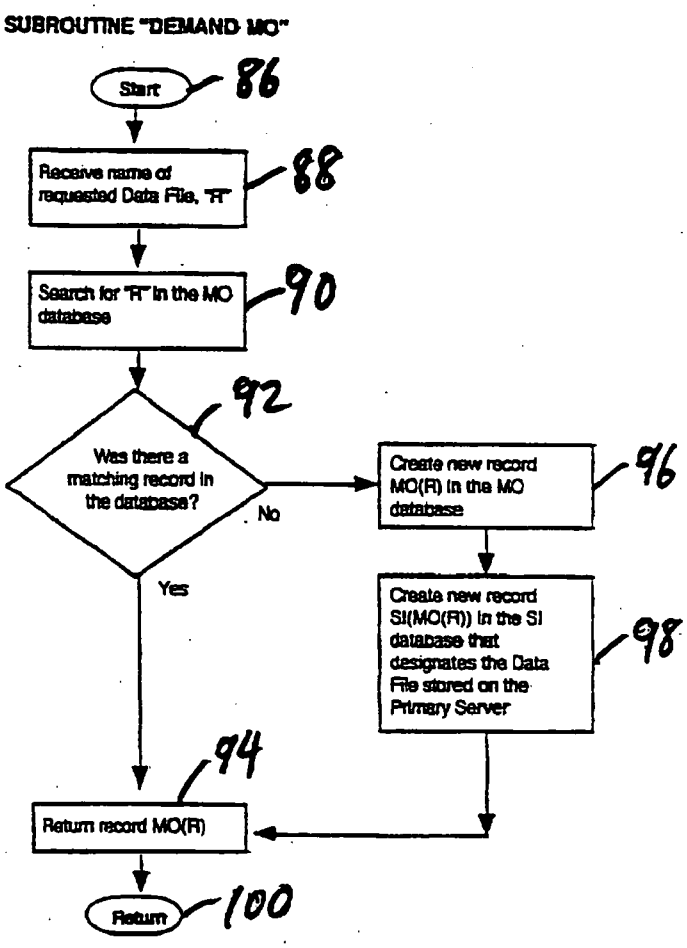
MAIN FLOW CHART FOR SOFTWARE FOR JAVELIN VT1 WEB SERVING SYSTEM

Fig. 2



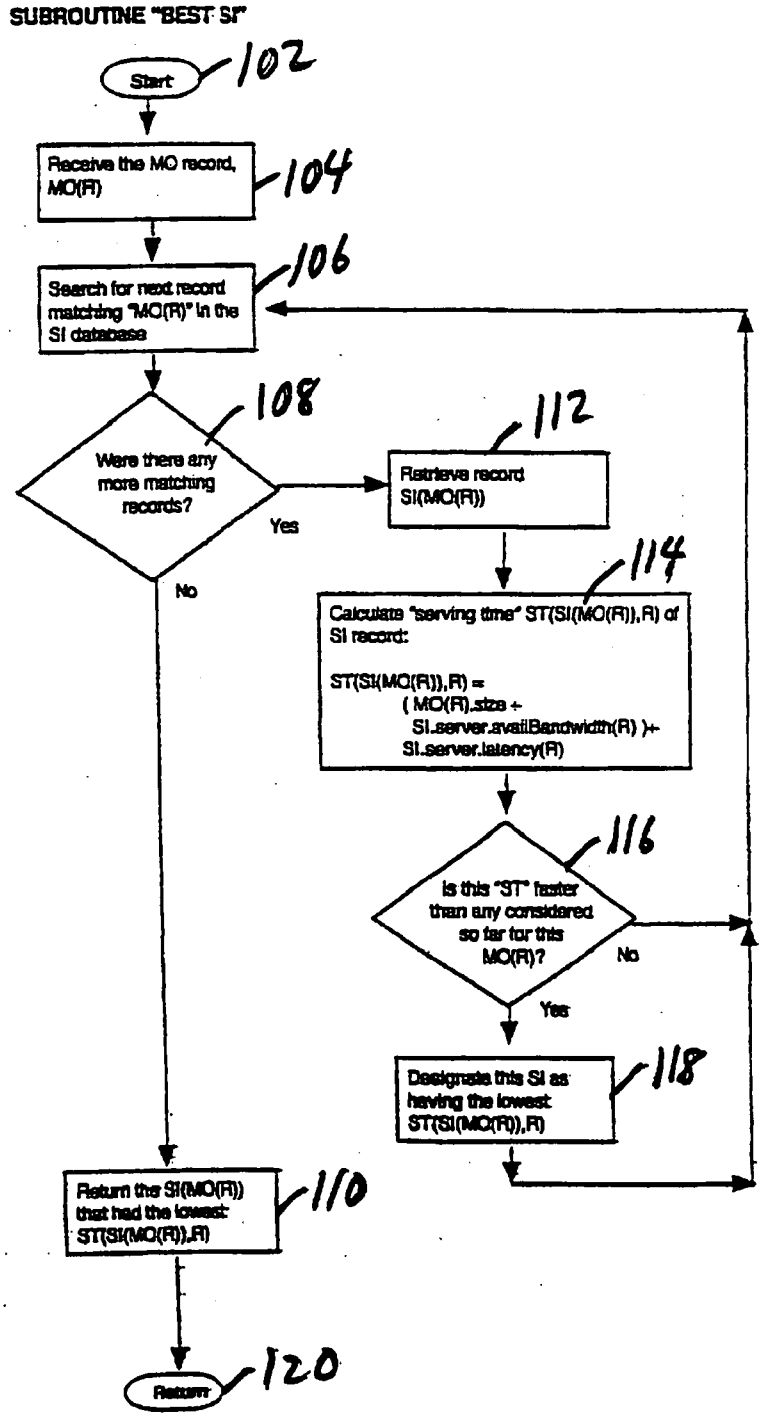
3/7

Fig. 3


SUBROUTINE "DEMAND MO"
Start
86
Receive name of requested Data File, "R"
88
Search for "R" in the MO database
90
Was there a matching record in the database?
92
No
Yes
Create new record MO(R) in the MO database
96
Create new record SI(MO(R)) in the SI database that designates the Data File stored on the Primary Server
98
Return record MO(R)
94
Return
100

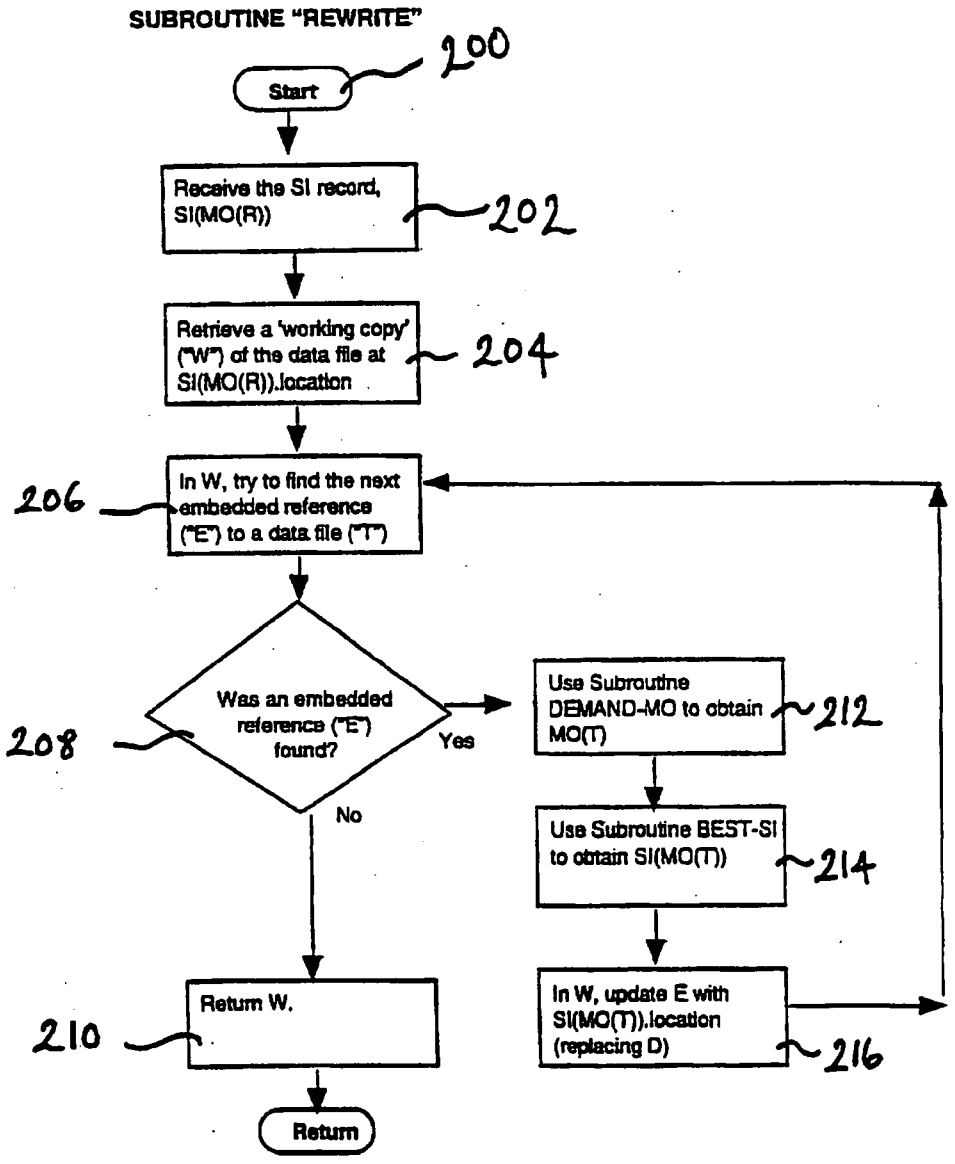
4/7

Fig. 4


SUBROUTINE "BEST SI"
Start
102
Receive the MO record, MO(R)
104
Search for next record matching "MO(R)" in the SI database
106
Were there any more matching records?
108
Yes
No
Retrieve record SI(MO(R))
112
Calculate "serving time" ST(SI(MO(R)),R) of SI record:
ST(SI(MO(R)),R) = (MO(R).size ÷ SI.server.availBandwidth(R))+ SI.server.latency(R)
114
Is this "ST" faster than any considered so far for this MO(R)?
116
No
Yes
Designate this SI as having the lowest ST(SI(MO(R)),R)
118
Return the SI(MO(R)) that had the lowest ST(SI(MO(R)),R)
110
Return
120

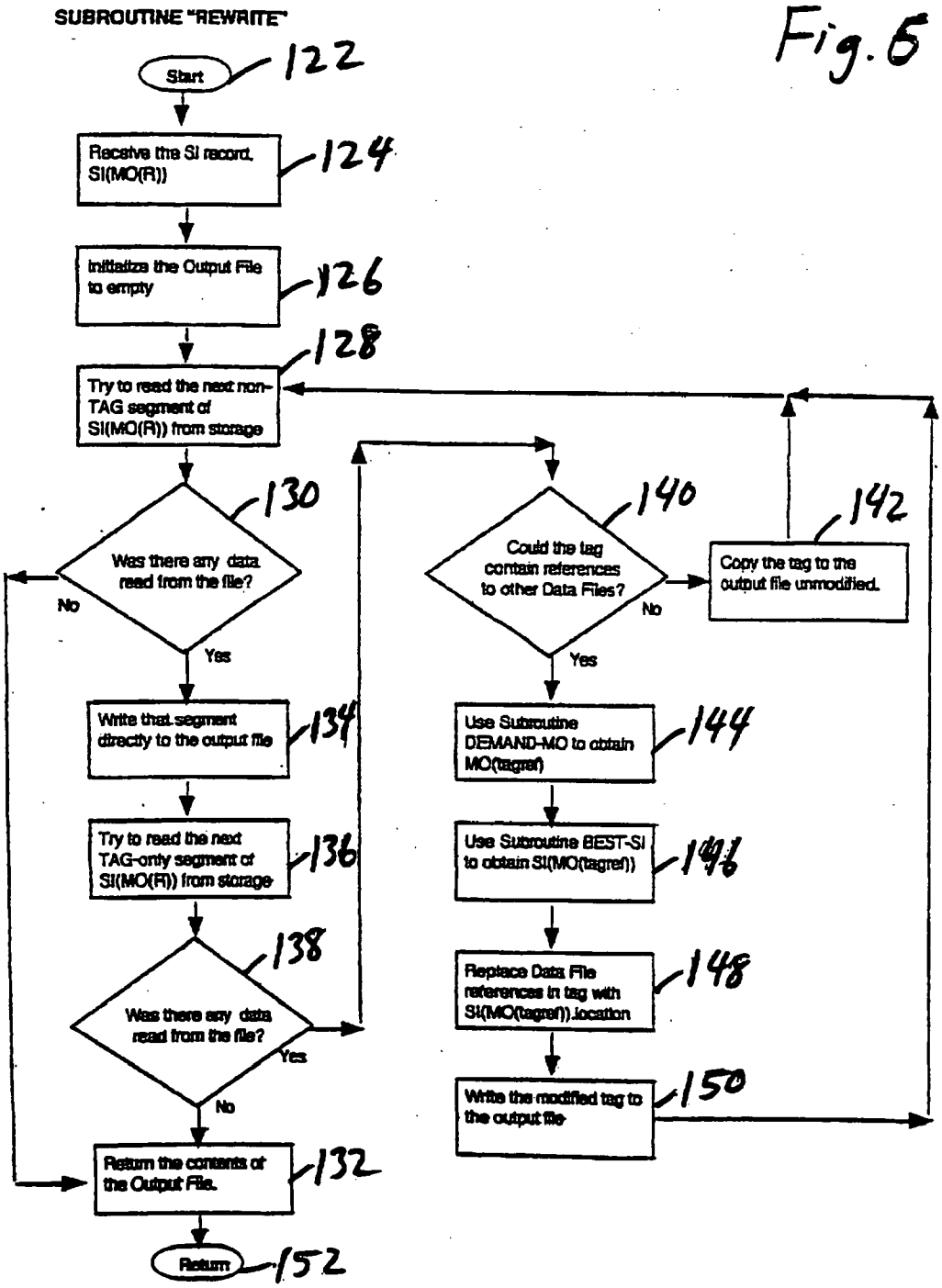
5/7

Fig. 5


SUBROUTINE "REWRITE"
Start
200
Receive the SI record, SI(MO(R))
202
Retrieve a 'working copy' ("W") of the data file at SI(MO(R)).location
204
206
In W, try to find the next embedded reference ("E") to a data file ("T")
208
Was an embedded reference ("E") found?
Yes
No
Use Subroutine DEMAND-MO to obtain MO(T)
212
Use Subroutine BEST-SI to obtain SI(MO(T))
214
In W, update E with SI(MO(T)).location (replacing D)
216
210
Return W.
Return

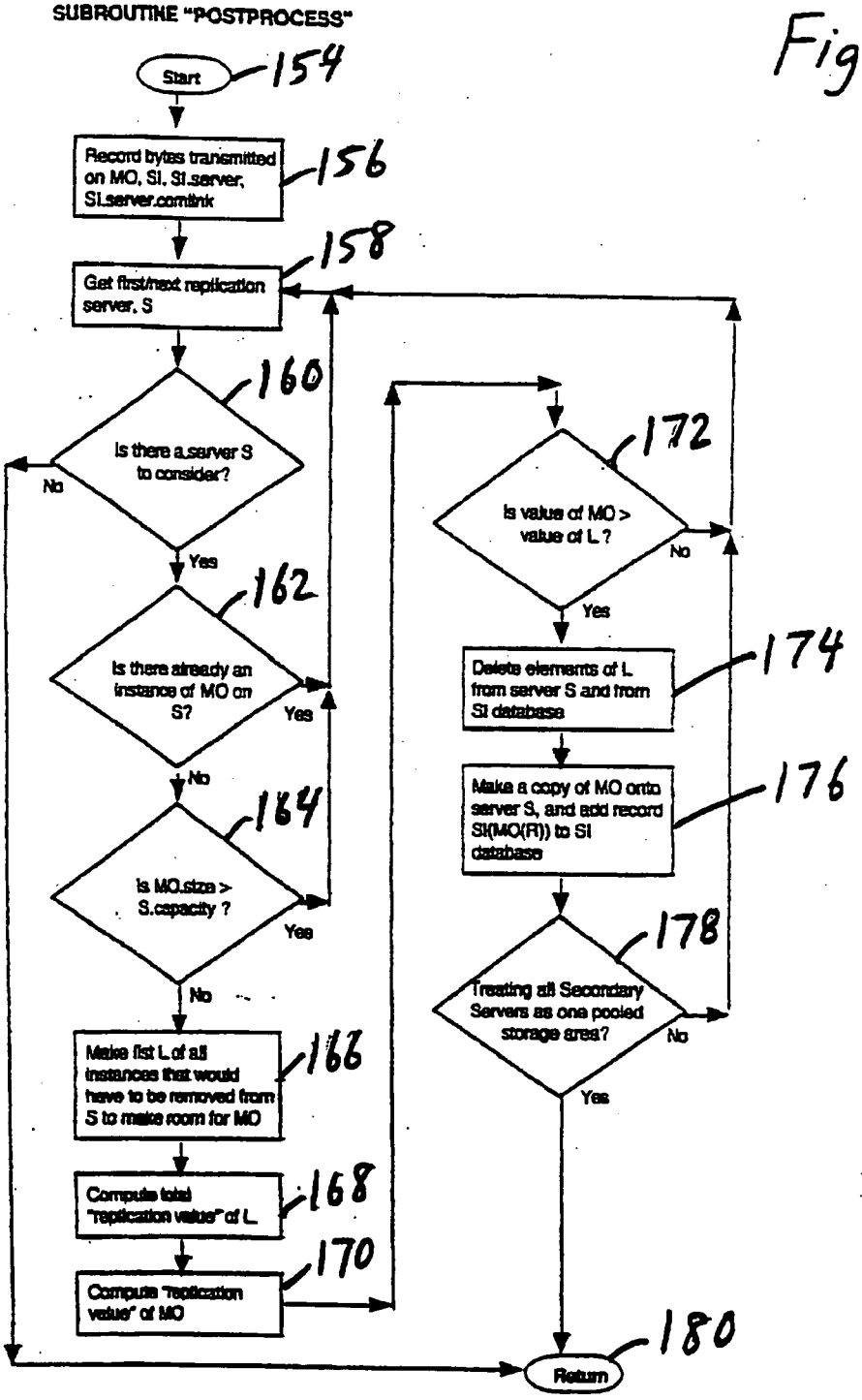
6/7

Fig. 6


SUBROUTINE "REWRITE"
Start
122
Receive the SI record, SI(MO(R))
124
Initialize the Output File to empty
126
Try to read the next non-TAG segment of SI(MO(R)) from storage
128
Was there any data read from the file?
130
No
Yes
Write that segment directly to the output file
134
Try to read the next TAG-only segment of SI(MO(R)) from storage
136
Was there any data read from the file?
138
Yes
No
Return the contents of the Output File.
132
Return
152
Could the tag contain references to other Data Files?
140
No
Copy the tag to the output file unmodified.
142
Yes
Use Subroutine DEMAND-MO to obtain MO(tagref)
144
Use Subroutine BEST-SI to obtain SI(MO(tagref))
146
Replace Data File references in tag with SI(MO(tagref)).location
148
Write the modified tag to the output file
150

7/7

Fig. 7


SUBROUTINE "POSTPROCESS"
Start
154
Record bytes transmitted on MO, SI, SI.server, SI.server.comlink
156
Get first/next replication server, S
158
Is there a server S to consider?
160
No
Yes
Is there already an instance of MO on S?
162
Yes
No
Is MO.size > S.capacity ?
164
Yes
No
Make list L of all instances that would have to be removed from S to make room for MO
166
Compute total "replication value" of L
168
Compute "replication value" of MO
170
Is value of MO > value of L ?
172
No
Yes
Delete elements of L from server S and from SI database
174
Make a copy of MO onto server S, and add record SI(MO(R)) to SI database
176
Treating all Secondary Servers as one pooled storage area?
178
No
Yes
Return
180

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/13097

A. CLASSIFICATION OF SUBJECT MATTER

IPC(6) :G06F 17/30; H04J 3/22

US CL :395/761,752,200.33,200.47,200.59

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 395/761,752,200.33,200.47,200.59

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

IEEE data base, Computer Select database

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y,E	US 5,659,729 A (NIELSEN) 19 August 1997, col. 3-7.	1-38
Y,P	US 5,572,643 A (JUDSON) 05 November 1996, col. 2-6.	1-38
Y,P	US 5,623,656 A (LYONS) 22 April 1997, col.2-4.	1-38
A,P	US 5,625,781 A (CLINE et al) 29 April 1997, col. 2-6.	1-38
A,E	US 5,666,362 A (CHEN et al) 09 September 1997, col. 2-11.	1-38
A,E	US 5,671,279 A (ELGAMAL) 23 September 1997, see the whole reference.	1-38

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
29 SEPTEMBER 1997	29 OCT 1997
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer ZARNI MAUNG Telephone No. (703) 305-3900

Form PCT/ISA/210 (second sheet)(July 1992)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US97/13097

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A,P	US 5,619,648 A (CANALE et al) 08 April 1997, see the whole reference.	1-38
A,P	US 5,649,186 A (FERGUSON) 15 July 1997, see the whole reference.	1-38

Form PCT/ISA/210 (continuation of second sheet)(July 1992)*

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 9/44, 17/30

A1

(11) International Publication Number: WO 98/06033

(43) International Publication Date: 12 February 1998 (12.02.98)

(21) International Application Number: PCT/US97/13817

(22) International Filing Date: 8 August 1997 (08.08.97)

(30) Priority Data:
60/023,373 8 August 1996 (08.08.96) US

(71) Applicant: AGRANAT SYSTEMS, INC. [US/US]; 1345 Main Street, Waltham, MA 02154 (US).

(72) Inventors: AGRANAT, Ian, D.; 108 Jericho Road, Weston, MA 02193 (US). GIUSTI, Kenneth, A.; 16 Josiah Drive, Upton, MA 01568 (US). LAWRENCE, Scott, D.; 560 Old Marlboro Road, Concord, MA 01742-4042 (US).

(74) Agent: ENGELSON, Gary, S.; Wolf, Greenfield & Sacks, P.C., 600 Atlantic Avenue, Boston, MA 02210 (US).

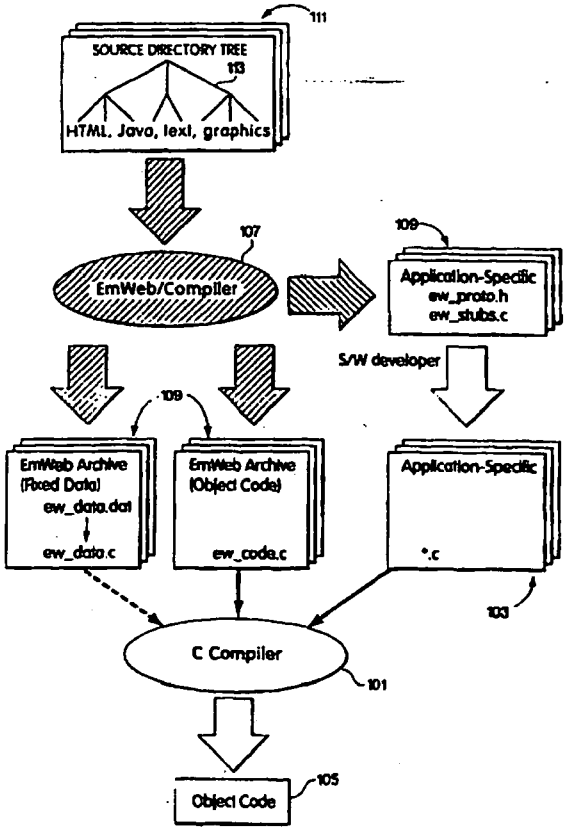
(81) Designated States: JP, European patent (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: EMBEDDED WEB SERVER

(57) Abstract

An embedded graphical user interface employs a World-Wide-Web communications and display paradigm. The development environment includes an HTML compiler which recognizes and processes a number of unique extensions to HTML. The HTML compiler produces an output which is in the source code language of an application to which the graphical user interface applies. A corresponding run-time environment includes a server which serves the compiled HTML documents to a browser.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

EMBEDDED WEB SERVER

COPYRIGHT NOTICE

The tables attached to the disclosure of this patent contain material that is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction by anyone of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

TABLES

Table A is a functional specification of the Agranat Systems, Inc. EmWeb™ product; and

Table B is a source code listing in the C programming language of header files defining data structures representing an archive of content as used by the EmWeb™ product.

The noted tables are provided to show an example of a particular embodiment of the invention incorporating features thereof which are described in detail below.

CROSS-REFERENCE TO RELATED APPLICATION

Priority is claimed under 35 U.S.C. §119(e) to the inventors' Provisional U.S. Patent Application Serial No. 08/023,373, entitled EXTENDED LANGUAGE COMPILER AND RUN TIME SERVER, filed August 8, 1996, now pending. The inventors' above-identified provisional U.S. patent application is incorporated herein by reference.

1. Field of the Invention

The present invention relates generally to graphical user interfaces (GUIs), i.e. user interfaces in which information can be presented in both textual form and graphical form. More particularly, the invention relates to GUIs used to control, manage, configure, monitor and diagnose software and hardware applications, devices and equipment using a World-Wide-Web client/server communications model. Yet more particularly, the invention relates to methods and apparatus for developing and using such GUIs based on a World-Wide-Web client/server communications model.

2. Related Art

Many modern communications, entertainment and other electronic devices require or could benefit from improved local or remote control, management, configuration, monitoring and diagnosing. It is common for such devices to be controlled by a software application program specifically written for each device. The design of such a device includes any hardware and operating environment software needed to support the application, which is then referred to as an embedded application, because it is embedded within the device. Embedded application programs are generally written in a high-level programming language such as C, C++, etc., referred to herein as a native application programming language. Other languages suitable to particular uses may also be employed. The application program communicates with users through a user interface, generally written in the same high-level language as the application.

The representation of an application in a native application programming language is referred to as the application program source code. A corresponding representation which can be executed on a processor is referred to as an executable image.

Before an application written in a high-level language can be executed it must be compiled and linked to transform the application source code into an executable image. A compiler receives as an input a file containing the application source code and produces as an output a file in a format referred to as object code. Finally, one or more object code files are linked to form the executable image. Linking resolves references an object module may make outside of that object module, such as addresses, symbols or functions defined elsewhere.

Source code may also define arrangements by which data can be stored in memory and conveniently referred to symbolically. Such defined arrangements are referred to as data structures because they represent the physical arrangement of data within memory, i.e., the structure into which the data is organized.

Most commonly, remote control, management, configuration, monitoring and diagnosing applications employ unique proprietary user interfaces integrated with the application software and embedded into the device. Frequently these user interfaces present and receive information in text form only. Moreover, they are not portable, generally being designed to operate on a specific platform, i.e., combination of hardware and software. The devices for which control, management, configuration and diagnosing are desired have only limited run-time resources available, such as memory and long-term storage space. Proprietary interfaces are frequently designed with such limitations to data presentation, data acquisition and portability because of

the development costs incurred in providing such features and in order to keep the size and run-time resource requirements of the user interface to a minimum. Since each user interface tends to be unique to the particular remote control, management, configuration, monitoring or diagnosing function desired, as well as unique to the operating system, application and hardware platform upon which these operations are performed, significant time and/or other resources may be expended in development. Graphics handling and portability have therefore been considered luxuries too expensive for most applications.

However, as the range of products available requiring control, management, configuration, monitoring or diagnosing increase, such former luxuries as graphical presentation and portability of the interface from platform to platform have migrated from the category of luxuries to that of necessities. It is well known that information presented graphically is more quickly and easily assimilated than the same information presented as text. It is also well known that a consistent user interface presented by a variety of platforms is more likely to be understood and properly used than unique proprietary user interfaces presented by each individual platform. Therefore, portable GUIs with low run-time resource requirements are highly desirable.

With the growing popularity and expansion of the Internet, one extremely popular public network for communications between computer systems, and development of the World-Wide-Web communication and presentation model, a new paradigm for communication of information has emerged.

The World-Wide-Web and similar private architectures such as internal corporate LANs. provide a "web" of interconnected document objects. On the World-Wide-Web, these document objects are located on various sites on the global Internet. The World-Wide-Web is also described in "The World-Wide Web, " by T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret, Communications of the ACM, 37 (8), pp. 76-82, August 1994, and in "World Wide Web: The Information Universe," by Berners-Lee, T., et al., in Electronic Networking: Research, Applications and Policy, Vol. 1, No. 2, Meckler, Westport, Conn., Spring 1992. On the Internet, the World-Wide-Web is a collection of documents (i.e., content), client software (i.e., browsers) and server software (i.e., servers) which cooperate to present and receive information from users. The World-Wide-Web is also used to connect users through the content to a variety of databases and services from which information may be obtained. However, except as explained below, the World-Wide-Web is based principally on static information contained in the content documents available to the browsers through the servers. Such a limitation would

make the World-Wide-Web paradigm useless as a GUI which must present dynamic information generated by a device or application.

The World-Wide-Web communications paradigm is based on a conventional client-server model. Content is held in documents accessible to servers. Clients can request, through an interconnect system, documents which are then served to the clients through the interconnect system. The client software is responsible for interpreting the contents of the document served, if necessary.

Among the types of document objects in a "web" are documents and scripts. Documents in the World-Wide-Web may contain text, images, video, sound or other information sought to be presented, in undetermined formats known to browsers or extensions used with browsers. The presentation obtained or other actions performed when a browser requests a document from a server is usually determined by text contained in a document which is written in Hypertext Mark-up Language (HTML). HTML is described in HyperText Markup Language Specification - 2.0, by T. Berners-Lee and D. Connolly, RFC 1866, proposed standard, November 1995, and in "World Wide Web & HTML," by Douglas C. McArthur, in Dr. Dobbs Journal, December 1994, pp. 18-20, 22, 24, 26 and 86. HTML documents stored as such are generally static, that is, the contents do not change over time except when the document is manually modified. Scripts are programs that can generate HTML documents when executed.

HTML is one of a family of computer languages referred to as mark-up languages. Mark-up languages are computer languages which describe how to display, print, etc. a text document in a device-independent way. The description takes the form of textual tags indicating a format to be applied or other action to be taken relative to document text. The tags are usually unique character strings having defined meanings in the mark-up language. Tags are described in greater detail, below.

HTML is used in the World-Wide-Web because it is designed for writing hypertext documents. The formal definition is that HTML documents are Standard Generalized Markup Language (SGML) documents that conform to a particular Document Type Definition (DTD). An HTML document includes a hierarchical set of markup elements, where most elements have a start tag, followed by content, followed by an end tag. The content is a combination of text and nested markup elements. Tags are enclosed in angle brackets ('<' and '>') and indicate how the document is structured and how to display the document, as well as destinations and labels for hypertext links. There are tags for markup elements such as titles, headers, text attributes such as

bold and italic, lists, paragraph boundaries, links to other documents or other parts of the same document, in-line graphic images, and many other features.

For example, here are several lines of HTML:

```
Some words are <B>bold</B>, others are <I>italic</I>. Here we start a new
paragraph.<P>Here's a link to
the <A HREF="http://www.agranat.com">Agranat Systems, Inc.</A> home
page.
```

This sample document is a hypertext document because it contains a "link" to another document, as provided by the "HREF=." The format of this link will be described below. A hypertext document may also have a link to other parts of the same document. Linked documents may generally be located anywhere on the Internet. When a user is viewing the document using a client program called a Web browser (described below), the links are displayed as highlighted words or phrases. For example, using a Web browser, the sample document above would be displayed on the user's screen as follows:

Some words are **bold**, others are *italic*. Here we start a new paragraph.
Here's a link to Agranat Systems, Inc. home page.

In the Web browser, the link may be selected, for example by clicking on the highlighted area with a mouse. Selecting a link will cause the associated document to be displayed. Thus, clicking on the highlighted text "Agranat Systems, Inc." would display that home page.

Although a browser can be used to directly request images, video, sound, etc. from a server, more usually an HTML document which controls the presentation of information served to the browser by the server is requested. However, except as noted below, the contents of an HTML file are static, i.e., the browser can only present a passive snapshot of the contents at the time the document is served. In order to present dynamic information, i.e., generated by an application or device, or obtain from the user data which has been inserted into an HTML-generated form, conventional World-Wide-Web servers use a "raw" interface, such as the common gateway interface (CGI), explained below. HTML provides no mechanism for presenting dynamic information generated by an application or device, except through a raw interface, such as the CGI. Regarding obtaining data from the user for use by the application or device, although standard HTML provides a set of tags which implement a convenient mechanism for serving interactive forms to the browser, complete with text fields, check boxes

and pull-down menus, the CGI must be used to process submitted forms. Form processing is important to remote control, management, configuration, monitoring and diagnosing applications because forms processing is a convenient way to configure an application according to user input using the World-Wide-Web communications model. But, form processing using a CGI is extremely complex, as will be seen below, requiring an application designer to learn and implement an unfamiliar interface. A CGI is therefore not a suitable interface for rapid development and prototyping of new GUI capabilities. Moreover, a developer must then master a native application source code language (e.g., C, C++, etc.), HTML and the CGI, in order to develop a complete application along with its user interface.

Models of the World-Wide-Web communications paradigm for static content and dynamic content are shown in Figs. 14 and 15, respectively. As shown in Fig. 14, a browser 1401 makes a connection 1402 with a server 1403, which serves static content 1405 from a storage device 1407 to the browser 1401. In the case of dynamic content, shown in Fig. 15, the server 1403 passes control of the connection 1402 with the browser 1401 to an application 1501, through the CGI 1503. The application 1501 must maintain the connection 1402 with the browser 1401 and must pass control back to the server 1403 when service of the request which included dynamic content is complete. Furthermore, during service of a request which includes dynamic content, the application 1501 is responsible for functions normally performed by the server 1403, including maintaining the connection 1402 with the browser 1401, generating headers in the server/browser transport protocol, generating all of the static and dynamic content elements, and parsing any form data returned by the user. Since use of the CGI 1503 or other raw interface forces the application designer to do all of this work, applications 1501 to which forms are submitted are necessarily complex.

In order to provide dynamic content to a browser, the World-Wide-Web has also evolved to include Java and other client side scripting languages, as well as some server side scripting languages. However, these languages are interpreted by an interpreter built into the browser 1401 or server 1403, slowing down the presentation of information so generated. In the case of client side scripting, the script does not have any direct access to the application or to application specific information. Therefore, in order to generate or receive application specific information using client side scripting, the CGI 1503 or other raw interface must still be used. In the case of server side scripting, the server 1403 must parse the content as it is served, looking for a script to be interpreted. The access which a script has to the application is limited by the definition of the

scripting language, rather than by an application software interface designed by the application designer.

A server side script is an executable program, or a set of commands stored in a file, that can be run by a server program to produce an HTML document that is then returned to the Web browser. Typical script actions include running library routines or other applications to get information from a file, a database or a device, or initiating a request to get information from another machine, or retrieving a document corresponding to a selected hypertext link. A script may be run on the Web server when, for example, the end user selects a particular hypertext link in the Web browser, or submits an HTML form request. Scripts are usually written in an interpreted language such as Basic, Practical Extraction and Report Language (Perl) or Tool Control Language (Tcl) or one of the Unix operating system shell languages, but they also may be written in programming languages such as the "C" programming language and then compiled into an executable program. Programming in Tcl is described in more detail in Tcl and the Tk Toolkit, by John K. Ousterhout, Addison-Wesley, Reading, MA, USA, 1994. Perl is described in more detail in Programming Perl, by Larry Wall and Randal L. Schwartz, O'Reilly & Associates, Inc., Sebastopol, CA, USA, 1992.

Each document object in a web has an identifier called a Universal Resource Identifier (URI). These identifiers are described in more detail in T. Berners-Lee, "Universal Resource Identifiers in World-Wide-Web: A Unifying Syntax for the Expression of Names and Addresses of Objects on the Network as used in the World-Wide Web," RFC 1630, CERN, June 1994; and T. Berners-Lee, L. Masinter, and M. McCahill, "Uniform Resource Locators (URL)," RFC 1738, CERN, Xerox PARC, University of Minnesota, December 1994. A URI allows any object on the Internet to be referred to by name or address, such as in a link in an HTML document as shown above. There are two types of URIs: a Universal Resource Name (URN) and a Uniform Resource Locator (URL). A URN references an object by name within a given name space. The Internet community has not yet fully defined the syntax and usage of URNs. A URL references an object by defining an access algorithm using network protocols. An example URL is "http://www.agranat.com" A URL has the syntax "scheme: scheme_specific_components" where

"scheme" identifies the access protocol (such as HTTP, FTP or GOPHER). For a scheme of HTTP, the URL may be of the form "http://host:port/path?search" where

"host" is the Internet domain name of the machine that supports the protocol;

"port" is the transmission control protocol (TCP) port number of the appropriate server (if different from the default);

"path" is a scheme-specific identification of the object; and

"search" contains optional parameters for querying the content of the object.

URLs are also used by web servers and browsers on private computer systems or networks and not just the World-Wide-Web.

A site, i.e. an organization having a computer connected to a network, that wishes to make documents available to network users is called a "Web site" and must run a "Web server" program to provide access to the documents. A Web server program is a computer program that allows a computer on the network to make documents available to the rest of the World-Wide-Web or a private web. The documents are often hypertext documents in the HTML language, but may be other types of document objects as well, as well as images, audio and video information. The information that is managed by the Web server includes hypertext documents that are stored on the server or are dynamically generated by scripts on the Web server. Several Web server software packages exist, such as the Conseil Europeen pour la Recherche Nucleaire (CERN, the European Laboratory for Particle Physics) server or the National Center for Supercomputing Applications (NCSA) server. Web servers have been implemented for several different platforms, including the Sun Sparc 11 workstation running the Unix operating system, and personal computers with the Intel Pentium processor running the Microsoft® MS-DOS operating system and the Microsoft® Windows™ operating environment.

Web servers also have a standard interface for running external programs, called the Common Gateway Interface (CGI). CGI is described in more detail in How To Set Up And Maintain A Web Site, by Lincoln D. Stein, Addison-Wesley, August 1995. A gateway is a program that handles incoming information requests and returns the appropriate document or generates a document dynamically. For example, a gateway might receive queries, look up the answer in an SQL database, and translate the response into a page of HTML so that the server can send the result to the client. A gateway program may be written in a language such as "C" or in a scripting language such as Perl or Tcl or one of the Unix operating system shell languages. The CGI standard specifies how the script or application receives input and parameters, and specifies how any output should be formatted and returned to the server.

A user (typically using a machine other than the machine used by the Web server) that wishes to access documents available on the network at a Web site must run a client program

called a "Web browser." The browser program allows the user to retrieve and display documents from Web servers. Some of the popular Web browser programs are: the Navigator browser from NetScape Communications Corp., of Mountain View, California; the Mosaic browser from the National Center for Supercomputing Applications (NCSA); the WinWeb browser, from Microelectronics and Computer Technology Corp. of Austin, Texas; and the Internet Explorer, from Microsoft Corporation of Redmond, Washington. Browsers exist for many platforms, including personal computers with the Intel Pentium processor running the Microsoft® MS-DOS operating system and the Microsoft® Windows™ environment, and Apple Macintosh personal computers.

The Web server and the Web browser communicate using the Hypertext Transfer Protocol (HTTP) message protocol and the underlying transmission control protocol/internet protocol (TCP/IP) data transport protocol of the Internet. HTTP is described in Hypertext Transfer Protocol - HTTP/1.0, by T. Berners-Lee, R. T. Fielding, H. Frystyk Nielsen, Internet Draft Document, October 14, 1995, and is currently in the standardization process. At this writing, the latest version is found in RFC Z068 which is a draft definition of HTTP/1.1. In HTTP, the Web browser establishes a connection to a Web server and sends an HTTP request message to the server. In response to an HTTP request message, the Web server checks for authorization, performs any requested action and returns an HTTP response message containing an HTML document resulting from the requested action, or an error message. The returned HTML document may simply be a file stored on the Web server, or it may be created dynamically using a script called in response to the HTTP request message. For instance, to retrieve a document, a Web browser sends an HTTP request message to the indicated Web server, requesting a document by its URL. The Web server then retrieves the document and returns it in an HTTP response message to the Web browser. If the document has hypertext links, then the user may again select a link to request that a new document be retrieved and displayed. As another example, a user may fill in a form requesting a database search, the Web browser will send an HTTP request message to the Web server including the name of the database to be searched and the search parameters and the URL of the search script. The Web server calls a program or script, passing in the search parameters. The program examines the parameters and attempts to answer the query, perhaps by sending a query to a database interface. When the program receives the results of the query, it constructs an HTML document that is

returned to the Web server, which then sends it to the Web browser in an HTTP response message.

Request messages in HTTP contain a "method name" indicating the type of action to be performed by the server, a URL indicating a target object (either document or script) on the Web server, and other control information. Response messages contain a status line, server information, and possible data content. The Multipurpose Internet Mail Extensions (MIME) are a standardized way for describing the content of messages that are passed over a network. HTTP request and response messages use MIME header lines to indicate the format of the message. MIME is described in more detail in MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies, Internet RFC 1341, June 1992.

The request methods defined in the HTTP/1.1 protocol include GET, POST, PUT, HEAD, DELETE, LINK, and UNLINK. PUT, DELETE, LINK and UNLINK are less commonly used. The request methods expected to be defined in the final version of the HTTP/1.1 protocol include GET, POST, PUT, HEAD, DELETE, OPTIONS and TRACE. DELETE, PUT, OPTIONS and TRACE are expected to be less commonly used. All of the methods are described in more detail in the HTTP/1.0 and HTTP/1.1 specifications cited above.

Finally, a device or application using conventional World-Wide-Web technology must have access to a server. Conventional servers are large software packages which run on relatively large, resource-rich computer systems. These systems are resource-rich in terms of processing speed and power, long-term storage capacity, short-term storage capacity and operating system facilities. Conventional servers take advantage of these resources, for example, in how they store content source documents. For high-speed, convenient access to content, it is conventionally stored in a directory tree of bulky ASCII text files. Therefore, conventional World-Wide-Web technology cannot be used to implement a GUI in a relatively small, inexpensive, resource-poor device or application.

The combination of the Web server and Web browser communicating using an HTTP protocol over a computer network is referred to herein as the World-Wide-Web communications paradigm.

SUMMARY OF THE INVENTION

It is therefore an object of the present invention to provide an improved graphical user interface (GUI) for use in connection with remote control, management, configuration, monitoring and diagnosing functions embedded in applications, devices and equipment.

According to one aspect of the invention, there is provided a method for providing a graphical user interface having dynamic elements. The method begins by defining elements of the graphical user interface in at least one text document written in a mark-up language. Next, the method defines including at a location in the document a code tag containing a segment of application source code. The text document is then served to a client which interprets the mark-up language; and when the location is encountered, the client is served a sequence of characters derived from a result of executing a sequence of instructions represented by the segment of application source code. An embodiment of code tags illustrating their use is described in detail, later.

According to another aspect of the invention, there is another method for providing a graphical user interface having dynamic elements. This method also defines elements of the graphical user interface in at least one text document written in a mark-up language. Included in the document is a string identified by prototype tags. The text document is served to a prototyping client which interprets the mark-up language but does not recognize and does not display the prototype tag, but does display the string. An embodiment of prototype tags illustrating their use is described in detail, later.

According to yet another aspect of the invention, there is yet another method for providing a graphical user interface having dynamic elements. Elements of the graphical user interface are defined in at least one text document written in a mark-up language. Included at a location in the document is a code tag containing a segment of application source code. Also included in the document is a string identified by prototype tags. The text document is compiled into a content source, which is subsequently decompiled into a replica of the text document. The replica of the text document is served to a client which interprets the mark-up language; and when the location is encountered in the replica, the client is served a character stream generated by executing the segment of application source code.

Yet another aspect of the invention is a software product recorded on a medium. The software product includes a mark-up language compiler which can compile a mark-up language document into a data structure in a native application programming language, the compiler

recognizing one or more code tags which designate included text as a segment of application source code to be saved in a file for compilation by a compiler of the native application programming language.

Another aspect of the invention is a method for providing a graphical user interface having displayed forms for entry of data. The steps of this method include defining elements of the graphical user interface in at least one text document written in a mark-up language; naming in the document a data item requested of a user and used by an application written in a native application programming language; and compiling the text document into a content source including a data structure definition in the native application programming language for the named data item.

Yet another aspect of the invention may be practiced in a computer-based apparatus for developing a graphical user interface for an application, the apparatus including an editor which can manipulate a document written in a mark-up language and a viewer which can display a document written in the mark-up language. The apparatus further includes a mark-up language compiler which recognizes a code tag containing a source code fragment in a native application source code language, the code tag not otherwise part of the mark-up language, the compiler producing as an output a representation in the native application source code language of the document, including a copy of the source code fragment.

In accordance with another aspect of the invention, there is a method for developing and prototyping graphic user interfaces for an application. The method includes accessing an HTML file, encapsulating portions of said HTML and entering source code therein, producing a source module from said HTML with encapsulated portions, producing source code for a server, and cross compiling and linking said application, said source code module and said server thereby producing executable object code.

The invention, according to another aspect thereof, may be a data structure fixed in a computer readable medium, the data structure for use in a computer system including a client and a server in communication with each other. The data structure includes cross-compiled, stored and linked, HTML files with encapsulated portions containing executable code associated with said application, server code, and application code, wherein said executable code is run when the HTML file is served thereby providing real time dynamic data associated with said application.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings, in which like reference numerals denote like elements:

Fig. 1 is a block diagram of that aspect of the invention relating to development systems;

Fig. 2 is a block diagram of that aspect of the invention relating to an embedded system;

Fig. 3 is an HTML text fragment illustrating the use of an EMWEB_STRING tag;

Fig. 4 is another HTML text fragment illustrating another use of an EMWEB_STRING tag;

Fig. 5 is an HTML text fragment illustrating the use of an EMWEB_INCLUDE tag;

Fig. 6 is another HTML text fragment illustrating another use of an EMWEB_INCLUDE tag;

Fig. 7 is an HTML text fragment showing a use of the EMWEB_ITERATE attribute in connection with an EMWEB_STRING tag;

Fig. 8 is an HTML text fragment showing a use of the EMWEB_ITERATE attribute in connection with an EMWEB_INCLUDE tag;

Fig. 9 is an example of forms processing showing the relationship between the HTML source code for the form and the form output produced;

Fig. 10 is a block diagram of the data structure which defines the header for the data.dat archive file;

Fig. 11 is a state diagram of an embedded system illustrating dynamic content processing;

Fig. 12 is a state diagram of an embedded system illustrating forms processing;

Fig. 13 is a state diagram of an embedded system illustrating suspend/resume processing;

Fig. 14 is a block diagram illustrating conventional World-Wide-Web communication of static content between a server and a client; and

Fig. 15 is a block diagram illustrating conventional World-Wide-Web communication of dynamic content between a server and a client.

DETAILED DESCRIPTION

The present invention will be better understood upon reading the following detailed description in connection with the figures to which it refers.

Embodiments of various aspects of the invention are now described. First, a development environment is described in which application development and graphical user interface development are closely linked, yet require a low level of complexity compared to conventional

development of an application and GUI. Second, an operating environment is described in which the application, a server and GUI are tightly coupled, compact and flexible. In the described system a GUI having portability, low run-time resource requirements and using any of a wide variety of systems available to a user as a universal front end, i.e. the point of contact with the user is software with which the user is already familiar.

Development Environment

Fig. 1 illustrates a development environment according to one aspect of the invention. Not all components of the environment are shown, but those shown are identified in the following discussion.

Conventionally, an application development environment may include a source code editor, a compiler 101, a linker and a run-time environment in which to test and debug the application. It is expected that development environments in accordance with the invention include those components of a conventional development environment which a developer may find useful for developing an application. In the case of embedded applications, i.e., applications included within a device or larger application, the run-time environment includes the device or application in which the application is embedded, or a simulation or emulation thereof.

The compiler 101 takes source code 103 generated using the source code editor or from other sources and produces object code 105, which is later linked to form the executable image.

In addition to the conventional elements noted above, the described embodiment of a development environment according to the invention includes an HTML compiler 107 whose output 109 is in the source code language of the application under development. In addition, the development environment may include an HTML editor, an HTTP-compatible server for communicating with client software, i.e., browsers, and an HTTP-compatible browser.

The HTML editor is used to create and edit HTML documents 111 which define the look and feel of a GUI for the application. Numerous tools are now available for performing this task while requiring a minimal knowledge or no knowledge of HTML, for example, Microsoft® Front Page™. It is preferred that the HTML editor used permit entry of non-standard tags into the HTML document.

As will be seen in further detail, below, the server and browser are used to test a prototype GUI before it is fully integrated with the application or in the absence of the application. The browser should be capable of making a connection with the server using, for

example, a conventional connection protocol such as TCP/IP, as shown and described above in connection with Fig. 14. Other protocols or direct connections can also be used, as would be understood by those skilled in this art. While the browser and the server may be connected through a network such as the Internet, they need not be. For example, the server and client may run and connect on a single computer system.

Application development proceeds substantially in a conventional manner as known to software developers. The application development should include the design of a software interface through which data will be communicated into and out of the application. However, the software interface is not a GUI. Rather, the interface merely defines how other software can communicate with the application. For example, the interface may be a collection of function calls and global symbols which other software can use to communicate with the application. The application should be written in a high level language such as C, C++, etc. The application can be tested by compiling and linking it with prototype code that provides or receives information through the software interface, exercising those features of the application.

Meanwhile, a GUI for the application is designed as follows. The look and feel of the GUI are developed using the HTML editor, server and browser to create a set of content source documents 111 including at least one HTML document, which together define the look and feel of the GUI. This aspect of GUI development is conventional, proceeding as though the developer were developing a World-Wide-Web site.

At locations in one or more HTML documents where data obtained from the application is to be displayed, the author includes special tags, explained further below, which allow the HTML document to obtain from the application the required data, using the application software interface.

The content source documents 111 are stored conventionally in the form of one or more directory trees 113. The directory tree 113 containing the content which defines the GUI is then compiled using the HTML compiler 107, to produce an application source code language output 109 representing the content source documents in the directory tree. The source code elements 109 produced from the content source documents 111 in the directory tree 113, source code for an HTTP compatible server (not shown) and the application source code 103 are compiled into object code 105 and linked to form an executable image. The server may be supplied in the form of an object code library, ready for linking into the finished executable image. The executable image thus formed fully integrates the graphical user interface defined using familiar tools of

World-Wide-Web content development with the control and other functions defined using conventional application development tools.

In order to successfully perform the integration described above, the HTML compiler 107 of the described embodiment of the invention, the EmWeb™/compiler 107, recognizes a number of special extensions to HTML. The HTML extensions implemented by the EmWeb™/compiler 107, embodying aspects of the invention are described in detail in Table A, Section 3.2. Several of these extensions are described briefly here, to aid in understanding the invention.

The EMWEB_STRING tag is an extension of HTML used to encapsulate a fragment of source code in the HTML document. The source code will be executed by a system in which the application is embedded when the document is served to a browser (usually running on another system) and the location of the EMWEB_STRING tag is reached. The source code returns a character string that is inserted as is into the document at the location of the EMWEB_STRING tag. Examples of the use of the EMWEB_STRING tag are shown in Figs. 3 and 4.

In the example of Fig. 3, the EMWEB_STRING tag 301 first defines using "C=" a boundary character 303 used to define the end 305 of the included source code. Immediately following the boundary character definition is a fragment of C code 307 which returns a pointer to a string representing one of three fax states. When served by an embedded application, this example HTML produces the text "NetFax State:" followed by "Sending", "Receiving" or "Idle", depending on the value of the symbol GlobalFaxState.

The example of Fig. 4 shows the use of EMWEB_STRING to output typed data whose type is defined by an attribute, EMWEB_TYPE 401. The EmWeb™/compiler uses this attribute 401 to produce a source code output routine which converts the typed data found at the address returned 403 into a string for serving at the proper location in the document.

A similar function is performed by the HTML extension, the EMWEB_INCLUDE tag. Using this tag, standard parts of a GUI such as headers and footers common to multiple pages or windows of information need only be stored once. Header and footer files are referred to using the EMWEB_INCLUDE tag which inserts them at the location in each HTML content document where the tag is placed. In the described embodiment of the invention, the contents of the EMWEB_INCLUDE tag must resolve to a relative or absolute path name within the local directory tree of content. This can be done by specifying a local Universal Resource Locator (URL), which is how resources are located in the World-Wide-Web communications paradigm, or by including source code which returns a string representing such a local URL. An absolute

local URL takes the form "/path/filename", where "/path" is the full path from the root of the directory tree to the directory in which the file is located. A relative URL defines the location of a file relative to the directory in which the current, i.e., base, document is located and takes the form "path/filename". While the described embodiment requires resolution of the EMWEB_INCLUDE tag to a local URL, the invention is not so limited. In alternate embodiments, local and external URLs may be permitted or other limitations imposed. Examples of the use of the EMWEB_INCLUDE tag are shown in Figs. 5 and 6.

In the example of Fig. 5, a COMPONENT attribute 501 in an EMWEB_INCLUDE tag simply defines a local URL 503.

In the more elaborate example of Fig. 6, a fragment of source code 601 which produces a local URL 603 upon a defined condition 605 is used to generate a local URL at run time.

The results to be returned by an EMWEB_STRING or EMWEB_INCLUDE tag can also be built up iteratively using repeated calls to the included source code. This is done using the EMWEB_ITERATE attribute, yet another extension to HTML. Examples of the use of EMWEB_ITERATE are shown in Figs. 7 and 8.

Fig. 7 shows an example of the EMWEB_ITERATE attribute 701 used in connection with the EMWEB_STRING tag 703. The fragment of code 705 is executed repeatedly until a NULL is returned. Thus, this HTML repeatedly executes the C source code fragment to display the tray status of all trays in a system.

Similarly, in Fig. 8, EMWEB_INCLUDE 801 and EMWEB_ITERATE 803 are used to build a table of features for which content from other URLs 805 are to be displayed. When the table is complete, a NULL is returned 807, terminating the iterations.

Since the extensions to HTML described above allow the encapsulation of source code within an HTML document a mechanism with which to provide the encapsulated source code with required global definitions, header files, external declarations, etc. is also provided in the form of an EMWEB_HEAD tag. The EMWEB_HEAD tag specifies a source code component to be inserted in the source code output of the EmWeb™/compiler, outside of any defined function. Although it is preferred that the EMWEB_HEAD tag appears in the HTML file header, it may appear anywhere. The code generated by an EMWEB_HEAD tag is placed before any functions or other code defined within the HTML content source documents.

As indicated above, the GUI may be prototyped using a conventional server and browser (see Fig. 14) to preview the HTML documents comprising the GUI. Therefore, it may be useful

to provide static content with which to preview the page, at locations where dynamic content will appear during use, but which does not appear in the compiled document. For example, it may be useful to include a prototyping value for content which is otherwise provided using the EMWEB_STRING tag mechanism. Therefore, another extension to HTML recognized by the EmWeb™/compiler is the EMWEB_PROTO begin 309 and end 311 tags, as shown in Fig. 3. The EmWeb™/compiler removes these tags and everything between them when compiling the document, but the tags are ignored and the text between them is interpreted normally by a conventional browser viewing the HTML document either directly or via a conventional server. Conventional browsers recognize the tag due to its special syntax, e.g., being enclosed in "<" and ">", but are designed to ignore and not display any tag for which the browser does not have a definition. All EmWeb™/compiler HTML extensions are thus skipped over by conventional browsers. Thus, in the example of Fig. 3, the prototype page displays "NetFax State: Sending". Fig. 4 shows a similar use of EMWEB_PROTO tags.

Handling of HTML forms by the EmWeb™/compiler is now described in connection with Fig. 9. As seen in Fig. 9, an HTML form is defined substantially conventionally. Names used in the form are used in constructing symbol names used in the output source code produced by the EmWeb™/compiler. Therefore names should be valid symbol names in the source code language.

Each element of a form definition is translated by the EmWeb™/compiler into a part of a corresponding data structure defined for that form. Forms data is moved into and out of the application by changing values of items in the data structure.

Turning now to the example in Fig. 9, the relationship between the illustrated HTML form definition and the corresponding data structure is described. The form is given a unique name, using an EMWEB_NAME attribute in a FORM tag. The form name becomes part of the structure name, for easy reference and uniqueness. The form name will also be used to generate function names for functions which are called when the form is served and when the form is submitted.

The structure generated is itself composed of two structures. The first holds values of each dynamic element of the form. The second holds a status flag indicating the status of the contents of a corresponding value. Thus, in the example of Fig. 9, a structure to hold values and status for the sysName INPUT and the Logging SELECTion is created. The value of sysName is a character string, while Logging is an enumerated type.

Two function prototypes are also generated. The actions to be performed by these functions must be defined by the developer. The Serve function is called when the form is served and can be used to supply default values, for example. The Submit function is called when the form is submitted, to update values in the data structure, for example.

Currently, EmWeb™/compiler supports TEXT, PASSWORD, CHECKBOX, RADIO, IMAGE, HIDDEN, SUBMIT, RESET, SELECT and OPTION input fields. For detailed descriptions, see Table A, Section 3.2.5. In addition, the EmWeb™/compiler supports "typing" of TEXT input field data. That is, the EMWEB_TYPE attribute may be used to define a TEXT input field to contain various kinds of integers, a dotted IP address (i.e., an address of the form 000.000.000.000), various other address formats, etc. A mapping of EMWEB_TYPE values to C language types is formed in the table in Table A, Section 3.2.5.3.

The EmWeb™/compiler has been described in terms of a generic application source code language. The current commercial embodiment of the EmWeb™/compiler assumes the application source code language to be C or a superset thereof, e.g., C++. However, the functionality described can be generalized to any application source code language which may be preferred for a particular application purpose. However, in order to more fully understand how the EmWeb™/compiler and HTML extensions described above cooperate to permit integration of an HTML defined GUI with an application defined in an application source code, it will be assumed, without loss of generality, that the application source code language is C or a superset thereof.

The EmWeb™/compiler produces a set of output files including a data.dat file containing the fixed data of a content archive, a code.c file containing the generated source code portions of an archive including portions defined in EMWEB_STRING, EMWEB_INCLUDE and EMWEB_HEAD tags and other source code generated by the EmWeb™/compiler, as well as proto.h and stubs.c files containing the definitions of C functions used for forms processing. The structure of these files is now described in connection with the data structure illustrated in Fig. 10.

The content archive file data.dat has a header structure as illustrated in Fig. 10. The data structure is accessed through an archive header 1001 which is a table of offsets or pointers to other parts of the archive. For example, there is a pointer 1001a to a compression dictionary 1003 for archives which include compressed documents. There is also a pointer 1001b to a linked list of document headers 1005, 1007 and 1009. Each document header 1005, 1007 and

1009 is a table of offsets or pointers to various components of the document. For example, the document header includes a pointer 1005a to the URL 1011 to which the document corresponds. There is also a pointer 1005b to a field 1013 giving the Multipurpose Internet Mail Extension (MIME) type of the document. There are pointers 1005c and 1005d respectively to header nodes 1015 and document nodes 1017, explained further below. Finally, there is a pointer 1005e to a block of static compressed or uncompressed data 1019 representing the static portions of the document.

The static data does not include any EmWeb™ tags, i.e., the extensions to HTML discussed above and defined in detail in Table A. Rather, information concerning any EmWeb™ tags used in the document appears in the document nodes structure.

Each EmWeb™ tag employed in a document is represented in that document's document nodes structure as follows. The location of the EmWeb™ tag within an uncompressed data block or an uncompressed copy of a compressed data block is represented by an offset 1017a relative to the uncompressed data. The type of tag is indicated by a type flag 1017b. A node may include a flag which indicates any attributes associated with the tag represented. For example, a node for a tag of type EMWEB_STRING may include a flag indicating the attribute EMWEB_ITERATE. Finally, nodes include an index 1017c. In nodes defining form elements, the index holds a form number and element number uniquely identifying the element and form within the document. In nodes defining EMWEB_STRING tags, the index is a reference to the instance of source code which should be executed at that point. As such, the index may be evaluated in an expression of a "switch" statement in C, where each controlled statement of the "switch" statement is one source code fragment from one EMWEB_STRING instance. Alternatively, the index may be a pointer or index into a table of source code fragments from EMWEB_STRING tags, which have been encapsulated as private functions.

The data structure defined above provides a convenient way of transferring control as a document containing dynamic content is served. When a document is requested, the list of document nodes is obtained, to determine at what points control must be transferred to code segments which had been defined in the HTML source document. The document is then served using the data block defining the static elements of the document, until each document node is encountered. When each document node is encountered, control is transferred to the appropriate code segment. After the code segment completes execution, the static content which follows is served until the offset of the next document node is encountered.

Header nodes permit the storage of document meta information, not otherwise handled, such as content language, e.g., English, German, etc., cookie control, cache control or an e-tag giving a unique version number for a document, for example a 30-bit CRC value computed for the document. By avoiding having to put this information in the header of each document, significant space can be saved in the archive because not all documents require this information. Therefore, header nodes need only be stored for documents using this information.

The data structure which represents the archive of content used by the EmWeb™/compiler embodiment of the invention is defined by the C source code contained in Table B.

Run-time Environment

Aspects of the invention related to the run-time environment and server are embodied in the EmWeb™/server as described in detail in Table A, Section 4.

To a conventional browser implementing HTTP, the EmWeb™/server behaves conventionally. However, as shown in Fig. 2, the EmWeb™/server is fully integrated with the application and therefore has access to information about the application and device in which it is embedded.

Operation of the EmWeb™/server with respect to presentation of dynamic content is now described in connection with Fig. 11.

Before the operations shown in Fig. 11 commence, one or more archives are loaded by the server. When each archive is loaded, the server generates a hash table using the archive header data structure to make documents easy to locate using URLs.

First, the browser requests a document at a specified URL, using HTTP 1101. The EmWeb™/server acknowledges the request, in the conventional manner 1103. The EmWeb™/server then uses the hash table of the archive header to locate the document requested and begin serving static data from the document 1105. When a document node is encountered, for example denoting the presence of an EMWEB_STRING tag, then the server passes control to the code fragment 1107a of the application which had been included in the EMWEB_STRING tag 1107. When the code fragment completes execution and returns some dynamic data 1109, the EmWeb™/server then serves that dynamic data to the browser 1111. The EmWeb™/server then resumes serving any static data remaining in the document 1113. This process continues until the entire document, including all dynamic elements has been served.

Run-time serving and submission of forms is now described in connection with Fig. 12. A brief inspection of Fig. 12 will show that form service and submission proceeds along similar lines to those for serving dynamic content.

The browser first requests a URL using HTTP 1201. When, during service of the contents of the URL requested, a form is encountered, service of the form and any HTML-defined default values commences normally. The EmWeb™/server then makes a call to the application code 1203 to run a function 1203a which may substitute alternate default values 1205 with which to fill in the form. The document served then is made to include the default values defined by the static HTML as modified by the application software 1207. Later, when the user submits the form, the browser performs a POST to the URL using HTTP 1209. The form data is returned to the application by a call 1211 made by the EmWeb™/server to a function 1211 which inserts the data returned in the form into the data structure defined therefor within the application code. The response 1213 is then served back to the browser 1215.

Finally, it should be noted that there may be times when a request for dynamic content may require extended processing, unacceptably holding up or slowing down other operations performed by the application. In order to avoid such problems, the EmWeb™/server implements a suspend/resume protocol, as follows. The suspend/resume protocol exists within a context of a scheduler maintained and operated by the server. The scheduler includes a task list of scheduled server tasks to be performed.

Fig. 13 illustrates a situation where a browser requests a document containing an EMWEB_STRING tag whose processing is expected to interfere with other application operations. The initial HTTP request 1301 for the document is acknowledged 1303, conventionally. When the EMWEB_STRING tag is encountered, control transfers 1305a to the appropriate source code fragment 1305b in the application. The application then calls the suspend function 1307 of the EmWeb™/server and returns a dummy value 1309 to the function call generated at the EMWEB_STRING tag location. Calling the suspend function 1307 causes the scheduler to remove the EMWEB_STRING processing task from the task list. When the application has finally prepared the dynamic content required in the original function call, the application calls a resume function 1311 of the EmWeb™/server. Calling the resume function 1311 requeues the EMWEB-STRING processing task on the task list, as the current task. The EmWeb™/server responds by calling 1305c the function 1305d defined at the EMWEB_STRING tag again, this time immediately receiving a response from the application in

which the requested dynamic content 1313 is returned. The dynamic content is then served to the browser 1315.

The suspend/resume feature is particularly useful in distributed processing environments. If an embedded application is running on one processor of a distributed environment, but dynamic content can be requested which is obtained only from another processor or device in the distributed environment, then the use of suspend/resume can avoid lockups or degraded processing due to the need to obtain the dynamic content through a communication path of the distributed environment. Consider, for example, a distributed system including a control or management processor, and several communication devices. An embedded application running on the management processor can be queried for configuration data of any of the communication devices. Without suspend/resume, obtaining that data would tie up the communication path used by the management processor for control of the various communication devices, degrading performance.

The described embodiment of the invention illustrates several advantages thereof. For example, an embedded application can now have a GUI which is independent of either the application platform of that used to view the GUI. For example, the GUI can be operated through a Microsoft® Windows™ CE machine, Windows™ 3.x machine, Apple Macintosh, WebTV box, etc. running conventional browser software. Also, development of a GUI for an embedded application is greatly simplified. The look and feel is designed using conventional HTML design techniques, including straight-forward prototyping of the look and feel using a conventional client server system, using simple HTML extensions. Integration with the embedded application does not require the developer to learn or develop any special interface, but rather uses some HTML extensions to incorporate application source code directly into the HTML content. Yet another advantage in that the entire embedded application along with an HTTP-compatible server and the content to be served is reduced to a minimum of application source code, data structures for static data and data structures for dynamic data.

The present invention has now been described in connection with specific embodiments thereof. However, numerous modifications which are contemplated as falling within the scope of the present invention should now be apparent to those skilled in the art. For example, the invention is not limited to content whose source is HTML. Any mark up language could be used in the context of this invention. Alternatively, the content source could be raw text, which is particularly suitable for situations where the output of the user interface is also processed by one

or more automatic software text filters. Therefore, it is intended that the scope of the present invention be limited only by the properly construed scope of the claims appended hereto.

EmWeb™ Functional Specification

by

Ian Agranat

Ken Giusti

Scott Lawrence

6/20/97

Copyright © 1997 Agranat Systems, Inc. All Rights Reserved

Patent Pending

Agranat Systems Confidential

Table Of Contents

1. Introduction 1
1.1. Web-based Network Management 1
1.2. Web-based Systems, Devices, and Software 1
1.3. Goals 2
1.4. Assumptions 2

2. Overview 3

3. EmWeb/Compiler 4
3.1. EmWeb Context 6
3.2. HTML Extensions 6
3.2.1. EMWEB_STRING 6
3.2.2. EMWEB_INCLUDE 8
3.2.3. EMWEB_HEAD 9
3.2.4. EMWEB_PROTO 9
3.2.5. Forms 10
3.2.5.1. Form Element Names 10
3.2.5.2. Form Data Structure 11
3.2.5.3. HTML Form TEXT Input Fields 13
3.2.5.4. HTML Form PASSWORD Input Fields 14
3.2.5.5. HTML Form CHECKBOX Input Fields 15
3.2.5.6. HTML Form RADIO Input Fields 15
3.2.5.7. HTML Form IMAGE Input Fields 16
3.2.5.8. HTML Form HIDDEN Input Fields 16
3.2.5.9. HTML Form SUBMIT Input Fields 16
3.2.5.10. HTML Form RESET Input Fields 17
3.2.5.11. HTML Form SELECT/OPTION Fields 17
3.2.5.12. TEXTAREA 17
3.2.5.13. File Upload 18
3.2.6. Graphic Maps 18
3.2.7. Cache Control 19
3.2.8. EmWeb CGI 20
3.3. mime.types Configuration File 20
3.4. _ACCESS Files 22
3.5. Compiler Options 23
3.6. Compiler Output 25

4. EmWeb/Server 27
4.1. Application Interfaces 28
4.1.1. System Interfaces 29
4.1.1.1. Initialization and Shutdown 29
4.1.1.2. Scheduling 29
4.1.1.3. Memory Management 31

Copyright © 1997 Agranat Systems, Inc.

4.1.1.4. Time-of-Day Management 31
4.1.2. Network Interfaces 32
4.1.2.1. Network Buffer Management 32
4.1.2.2. TCP/IP Interfaces 33
4.1.3. Document and Archive Management 36
4.1.3.1. Installing and Removing Archives 36
4.1.3.2. Demand Loading 37
4.1.3.3. Cloning 38
4.1.3.4. URL Rewriting 38
4.1.3.5. Document Data Access 39
4.1.4. Authentication and Security 39
4.1.4.1. Basic Authentication 42
4.1.4.2. Manual Basic Authentication 43
4.1.4.3. Digest Authentication 44
4.1.4.4. Manual Digest Authentication 45
4.1.4.5. Application Security Verification 47
4.1.4.6. Document Realm Assignment 47
4.1.5. Request Context Access 48
4.1.6. Raw CGI 51
4.1.7. Logging Hook 53
4.1.8. Local Filesystem Interfaces 54
4.1.8.1. The EwsFileParams structure 54
4.1.8.2. File Upload 55
4.1.8.3. File Serve 58
4.2. Application Interface Examples 61
4.3. Porting Guidelines 65
4.3.1. Configuration Header Files 65
4.3.2. Application-Provided Functions 78
4.4. Memory Requirements 79

5. Conformance 85

6. Release History 86

7. References 87

Appendix A: EmWeb Application Interface Header Files 88
A-1. Configuration 88
A-1.1. src/config/ew_types.h 88
A-1.2. src/config/ew_config.h 91
A-2. Common Header Files 99
A-2.1. src/include/ews_api.h 99
A-2.2. src/include/ews_def.h 100
A-2.3. src/include/ews_sys.h 105
A-2.4. src/include/ews_net.h 112
A-2.5. src/include/ews_doc.h 118

Copyright © 1997 Agranat Systems, Inc.

A-2.6. src/include/ews_auth.h 123
A-2.7. src/include/ews_ctxt.h 130
A-2.8. src/include/ews_cgi.h 133

Copyright © 1997 Agranat Systems, Inc.

1. Introduction

The explosive growth of the World-Wide-Web in recent years has created a new standard for the Graphical User Interface (GUI) of networked applications: the Web browser. Netscape®, Mosaic, and other commercial and public-domain Web browsers are ubiquitous and inexpensive.

1.1. Web-based Network Management

Network management is a natural application for the World-Wide-Web. Before the Web, network management required end-users to purchase expensive custom-built GUI applications that used the SNMP protocol to manage network-based devices. With World-Wide-Web technology, many of these applications can be replaced by economical Web browsers.

World-Wide-Web standards make it possible to migrate the "look-and-feel" intelligence of the device configuration GUI away from third-party and custom-built network management applications and into the managed device itself. This migration gives vendors control over the configuration "look-and-feel" of their products among both low-cost Web browsers and the high-end third-party network management applications of the future.

Web-based network management can greatly accelerate the time-to-market for products traditionally managed by SNMP. With SNMP-based management, new product features require significant engineering resources to design and implement proprietary MIBs and develop customized GUI applications to manage the new features. On the other hand, Web-based management GUI applications can be rapidly prototyped and implemented using simple HTML forms.

SNMP will continue to play an important role in high-end network management applications by utilizing standard MIBs to monitor the overall topology and health of devices throughout the enterprise network. However, the new World-Wide-Web standards offer an exciting alternative to SNMP for the configuration and management of individual devices.

1.2. Web-based Systems, Devices, and Software

Embedded Web server technology can enable remote configuration, monitoring, and diagnostic capabilities for a wide range of applications. Imagine sending and receiving faxes, checking your voice-mail, and knowing the availability of your favorite candy bar in the vending machine down the hall by using your Web browser? Your office manager could configure the phone switch and voice-mail system, check the toner and paper levels in the photocopying machine, and monitor the security alarm system from their desktop. Your automobile mechanic could connect to your car's computer to perform engine diagnostics, and if they were stumped, the manufacturer could examine the results from around the world over the Internet.

Copyright © 1997 Agranat Systems, Inc.

1.3. Goals

EmWeb™ by Agranat Systems is a family of Embedded World-Wide-Web protocol software products engineered for the demanding real-time requirements of embedded systems.

Goals of the EmWeb product architecture include:

- Ease of use

  . Support rapid prototyping and implementation of new World Wide Web interfaces including device monitoring, configuration, and network management.
  . Easily portable to a wide-variety of real-time embedded operating environments with limited operating system capabilities, memory, and CPU resources.
  . Easily upgradable with new releases as World-Wide-Web standards continue to evolve rapidly.

- High performance

  . Efficient implementation for real-time processing environments.
  . Small foot-print to minimize FLASH and RAM resources required.

- Security

  . Support for authentication and security standards as they continue to evolve.
  . Configuration of method-independent access controls.

1.4. Assumptions

This functional specification assumes that the reader is familiar with the C programming language, the World-Wide-Web, and standards such as HTML (Hyper-Text Markup Language). Knowledge of HTTP (Hyper-Text Transport Protocol) and CGI (Common Gateway Interface) is also helpful to understand some of the more advanced features, but isn't required for most applications.

Copyright © 1997 Agranat Systems, Inc.

2. Overview

One draw-back to traditional World-Wide-Web servers is the CGI interface. Although HTML provides a convenient mechanism for sending interactive forms to a Web browser complete with text fields, checkboxes, and pull-down menus, the CGI interface used by server applications to process submitted forms is difficult to master. Form processing is essential for network management applications because it is the best way to implement a device configuration screen using World Wide Web protocols. However, CGI is not an appropriate interface for the rapid implementation and prototyping of new network management capabilities (EmWeb's number one goal). Another draw-back to traditional World-Wide-Web servers is the static nature of the content of files served to the Web browser. These files are typically resident on disk and are not expected to change frequently. On the other hand, the data of interest to network management applications is constantly changing (e.g. the number of packets received on a particular Ethernet port).

The EmWeb architecture works around the draw-backs of traditional World-Wide-Web servers by extending the standard HTML specification with new proprietary tags developed by Agranat Systems. These tags make it possible to provide a more intuitive HTML form processing interface for embedded systems, and give the application run-time control over the document content for the presentation of dynamic data. Note that these tags are not actually sent to the Web browser. Instead, they are stripped out and interpreted by the EmWeb/Compiler.

The EmWeb product consists of two components: The EmWeb/Compiler and the EmWeb/Server.

The EmWeb/Compiler is a tool provided by Agranat Systems that runs on an engineering development work-station under Unix, Windows/NT (x86), and Windows/95. The tool compiles one or more directories of files containing HTML, Java, text, graphics, etc., into one or more EmWeb archives. Each archive consists of an object code component and a fixed data component. These components are usually linked into the target's run-time image at compile time. However, the EmWeb architecture permits the dynamic loading and unloading of the fixed data components at run-time. Furthermore, if the target's operating system supports the dynamic loading and linking of object code modules, then entire archives may be loaded and unloaded at run-time.

The EmWeb/Server is a portable run-time library implemented in ANSI C which is compiled and linked with the target platform's application software. The EmWeb/Server implements the HTTP protocols to handle requests from Web browsers and interfaces with the archives created by the EmWeb/Compiler. The EmWeb/Server implements an application interface (API) to the target software environment's memory management, buffer management, and TCP/IP protocol stack.

Copyright © 1997 Agranat Systems, Inc.

3. EmWeb/Compiler

The EmWeb/Compiler processes directories of HTML and other Web documents to generate ANSI C code. The generated C code is then compiled using a C cross-compiler to generate object code for the target system. HTML forms are translated by the EmWeb/Compiler into a simple C function call interface customized to the application. The software developer is responsible for implementing the application-specific C code using these simple interfaces. The EmWeb/Compiler can generate function prototype definitions (a C include header file) to ensure correctness, and optionally generates stub functions to accelerate development.

The figure below illustrates the operation of the EmWeb/Compiler.

Copyright © 1997 Agranat Systems, Inc.

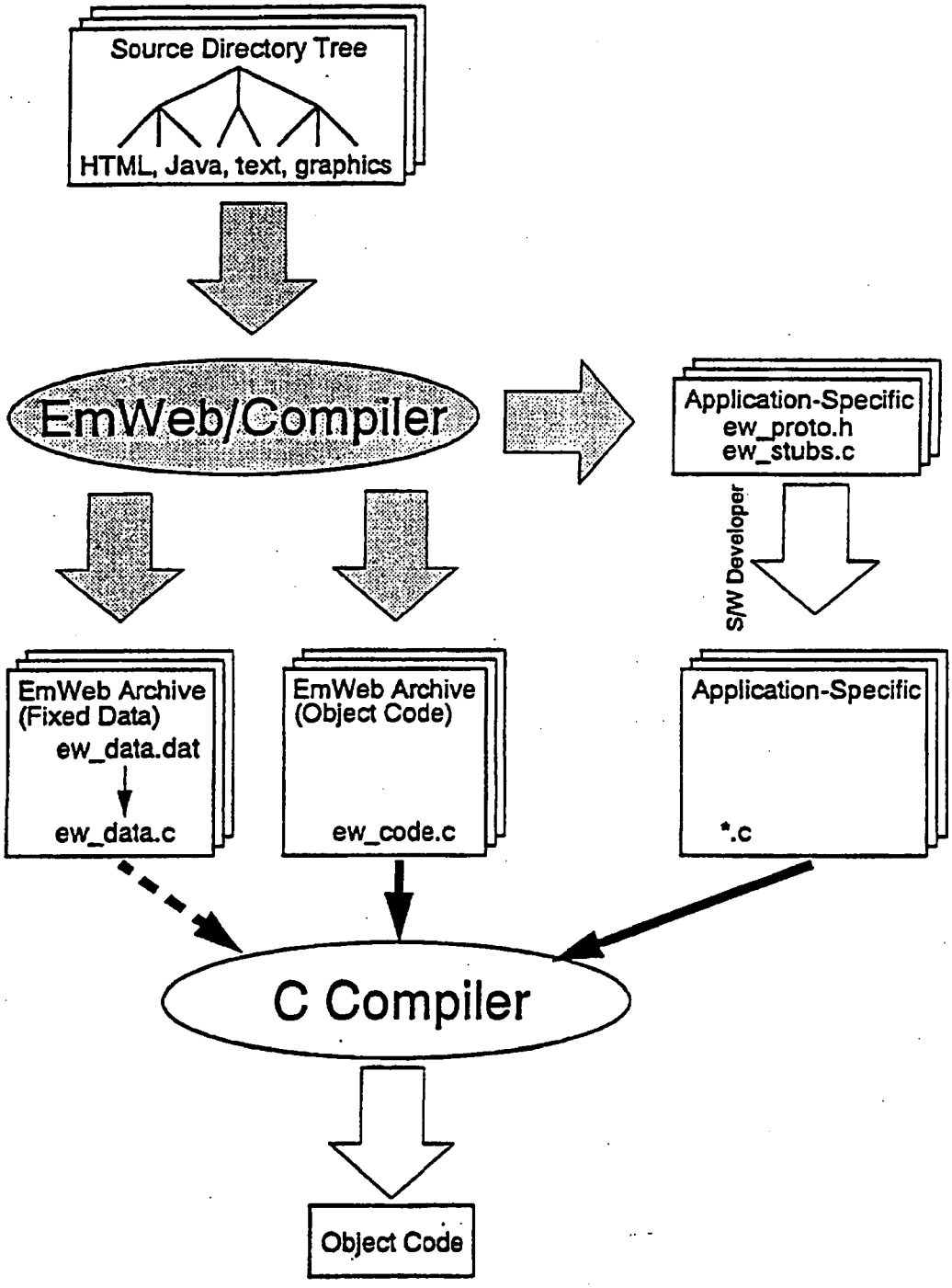


Figure 1: EmWeb/Compiler

3.1. EmWeb Context

The EmWeb/Server maintains a handle of type `EwsContext` at run-time that corresponds to a specific HTTP request from a Web browser client. This handle is made available to the application code responsible for providing the run-time content of HTML documents and forms.

The EmWeb/Server contains a library of functions available to the application for extracting information attached to this handle. For example, the application may want to determine the type of browser making a request, and change the look-and-feel of the requested document to take advantage of features available to particular browsers. In this example, the application could extract the browser type as follows:

```
bytes = ewsContextUserAgent (ewsContext,
        user_agent, sizeof(*user_agent));
```

For more information about available context information functions, refer to 4.1.5. Request Context Access, page 48.

3.2. HTML Extensions

The content to be provided by the EmWeb server is prepared as you would content for any normal Web server. The developer constructs a directory tree (or trees) of documents, which may include HTML documents, Java, text, graphics, and sound samples. The EmWeb compiler is then run over the content directory tree to build an internal database that is compiled and linked with the EmWeb/Server and the system code.

EmWeb defines proprietary HTML tags and attributes for access to system data and dynamic system control of the served documents. These tags are never served to browsers by the EmWeb/Server, and because (with one exception) they have no body part (they do not use a start and end tag, but only a stand-alone tag with attributes), they are ignored by browsers that see them when browsing the content tree either locally or via a non-EmWeb server. This facilitates testing and prototyping of content.

3.2.1. EMWEB_STRING

The `<EMWEB_STRING>` tag is used to provide applications with run-time control of an HTML document's content by encapsulating a fragment of C code in the document itself. This C code is executed as the document is being served to a Web browser and returns a null-terminated character string that is inserted as-is into the document in place of the `<EMWEB_STRING>` tag.

To include content in a document (which may include any standard HTML):

```
<EMWEB_STRING C='...; return (char *) somestring;'>
```

or

```
<EMWEB_STRING EMWEB_ITERATE C='...; return (char *) somestring;'>
```

Copyright © 1997 Agranat Systems, Inc.

The EmWeb/Compiler extracts the C language code fragment from the EMWEB_STRING tags in an HTML document and constructs a C module from them. Each fragment must end with a `return` statement that returns a pointer to a null-terminated string (or a NULL value which inserts no content). When the location of the EMWEB_STRING tag in the content is reached, the EmWeb/Server executes the code fragment and inserts the returned string into the document. If the optional EMWEB_ITERATE attribute is specified, the code fragment is called repeatedly until it returns a NULL pointer value. The number of iterations already processed (beginning with zero) is available to the code fragment by calling:

```
ewsContextIterations(ewsContext);
```

Note that the local variable

```
EwsContext ewsContext;
```

is made available to the EMWEB_STRING C code fragment by the EmWeb/Compiler.

The returned string must not be an automatic (stack) variable. The value is copied for transmission by EmWeb/Server immediately after returning. Thus, a single static string buffer may be shared by multiple code fragments. If pre-emptive multi-threading is to be used, the application may maintain a per-request buffer as part of its network handle. Alternatively, EMWEB_ITERATE may be used to support dynamic allocation of the string buffer as follows. On the first iteration (#0), memory may be allocated, filled with data, and returned to the EmWeb/Server for transmission. On the second iteration (#1), memory may be released and a NULL pointer returned to terminate the iteration.

The EmWeb/Compiler treats the embedded code opaquely; from the compiler's point of view, the code is simply a string of text that gets assembled into a generated code file. (See 3.6. Compiler Output, page 25).

As such, the EmWeb/Compiler needs to know the beginning and end of the embedded C code fragment. The character following the c= attribute in the EMWEB_STRING tag identifies the "bounds" of the embedded C code. This character must be either a single quote (') or double quote ("). When the EmWeb/Compiler identifies this "boundary character", it then treats all following characters as part of the embedded C code fragment, until it finds the next terminating "boundary character".

The obvious problem here is that the embedded C code can contain a single or double quote as part of the C code. For example:

```
return "Example of EmWeb's dynamic content\n";
```

In order to keep the EmWeb compiler from mistakenly confusing a single or double quote that belongs to the C code fragment as the ending "boundary character", the programmer must escape (using '\') any character intended for the C code that would be confused with the terminating boundary character. Thus, the above code fragment

Copyright © 1997 Agranat Systems, Inc.

could be embedded in an EMWEB_STRING tag in either of the following ways:

```
<EMWEB_STRING C='return "Example of EmWeb\'s dynamic content\n";'>
<EMWEB_STRING C="return \"Example of EmWeb's dynamic content\n\";">
```

The <EMWEB_STRING> tag may optionally be typed using the EMWEB_TYPE attribute as follows:

```
<EMWEB_STRING EMWEB_TYPE=DECIMAL_UINT C='
        return (uint32 *) &some_integer;'>
```

If EMWEB_TYPE is specified, then the C code fragment must return a pointer to data with the corresponding type. EmWeb/Server will convert the value into a character string automatically. A table of supported types can be found in section 3.2.5.3. HTML Form TEXT Input Fields, page 13

3.2.2. EMWEB_INCLUDE

The EmWeb/Server provides server-side inclusion of any part of its content tree using the EMWEB_INCLUDE tag; it is used in the same way as the EMWEB_STRING tag:

```
<EMWEB_INCLUDE COMPONENT='Local-URL'>
```

or

```
<EMWEB_INCLUDE C='...; return (char *) localURL;'>
```

or

```
<EMWEB_INCLUDE EMWEB_ITERATE C='...; return (char *) localURL;'>
```

If the COMPONENT tag is specified, the specified local element is resolved and inserted into the current document. This allows standard parts such as headers and footers to be stored only once.

As in EMWEB_STRING, the fragment specified by the C= attribute must end with a `return` statement that returns a pointer to a null-terminated string (or NULL). For EMWEB_INCLUDE, however, this string is treated as a relative or absolute path name within the local content tree. The path name is resolved to an entire document, which is then inserted in the current document for return to the browser. If the optional EMWEB_ITERATE attribute is specified, the code fragment is called repeatedly until it returns a NULL pointer value. The number of iterations already processed (beginning with zero) is available to the code fragment using the same access routine used for EMWEB_STRING. This tag is useful for building a response from components that may be part of the original document tree, or created dynamically.

Note that the local variable

```
EwsContext ewsContext;
```

is made available to the EMWEB_INCLUDE C code fragment by the EmWeb/Compiler.

Copyright © 1997 Agranat Systems, Inc.

The same quoting rules as EMWEB_STRING apply to EMWEB_INCLUDE.

3.2.3. EMWEB_HEAD

Because the <EMWEB_STRING> and <EMWEB_INCLUDE> tags allow the application to encapsulate fragments of C code within an HTML document, a mechanism is needed to provide the code fragments with the necessary global definitions, header files, external declarations, etc. needed for compilation. The <EMWEB_HEAD> tag is provided for this purpose.

```
<EMWEB_HEAD C='#include file-name.h'>
```

The EMWEB_HEAD tag, if present, should appear in the HEAD portion of the HTML document (this is not enforced, but the code fragments from all EMWEB_HEAD fragments are placed at the beginning of the generated module, so doing so is less confusing); it specifies a source fragment to be inserted in the generated module outside any C function. This fragment may declare variables, constants, or (as shown in the example above) C preprocessor directives - it could even define entire subroutines. This fragment is never executed directly by the EmWeb Server, so it does not insert any content into the document at the location of the tag.

Note that all EMWEB_HEAD code fragments from all HTML files in an archive are combined into the generated archive object code component source file. Therefore, it is recommended that common `#include` directives appear only in one document in the archive (typically, the index.html document at the archive's root directory).

The same quoting rules as EMWEB_STRING apply to EMWEB_HEAD.

3.2.4. EMWEB_PROTO

When designing an HTML document, it may occasionally be useful to provide content with which to preview the page, but which does not appear in the compiled document. This is accomplished with the EMWEB_PROTO begin and end tag; the EmWeb/Compiler removes these tags and everything between them when producing the document database, but the text between them is interpreted normally by a browser viewing the HTML document directly or via a conventional server. For example:

```
<H4>
System:  <EMWEB_STRING C="return mib2sysName;">
         <EMWEB_PROTO>system-name</EMWEB_PROTO>
</H4>
```

When compiled, the system name displayed in the first level header will be the value in the string variable `mib2sysName`, but viewed with a browser when previewing it looks like:

```
System: system-name
```

Note that the EmWeb/Compiler ignores HTML comments (e.g. "<!-- ... -->"). If EMWEB_STRING and EMWEB_INCLUDE tags are present within HTML comments, they will

Copyright © 1997 Agranat Systems, Inc.

be expanded by EmWeb/Server. If it is desirable to "comment out" an EMWEB_STRING or EMWEB_INCLUDE tag, EMWEB_PROTO tags should be used instead.

3.2.5. Forms

Form processing takes place as two operations: serving the form to the browser, possibly with current or default values set in some form input elements, and processing the values from the submitted form. Each of these operations is assisted by an application-provided routine associated with the form. Prototypes for these two application routines are produced for each form by the EmWeb/Compiler using information provided by EmWeb attributes on the FORM and its input elements.

> ***Restriction:*** *While HTML allows it, EmWeb does not support nested* FORM *tags (you may not have a form within a form).*

The elements for each form are combined in a generated C data structure by the EmWeb Compiler; this structure is passed by the EmWeb Server to each of the two form processing routines, and includes information on the status of each element.

3.2.5.1. Form Element Names

The EMWEB_NAME attribute is used to specify the names to be used by the EmWeb/Compiler when generating function and structure names.

On the FORM tag, the EMWEB_NAME attribute specifies a base name to be used to generate the function names and the form data structure name. (This implies that the value of the EMWEB_NAME attribute must be a valid C identifier). Note that if EMWEB_NAME is not present in a <FORM> tag, then the form is simply ignored by the EmWeb/Compiler and is passed as-is to the browser.

Consider the following example. Suppose there exists an HTML document "example.html" that contains:

```
<FORM METHOD=POST EMWEB_NAME=foo ACTION=bar >
```

The ACTION attribute is optional; if ACTION is not specified, then there may be only one form in the document. If ACTION is specified, it is served to the browser as "ACTION=example.html/bar". (This allows EmWeb/Server to differentiate between multiple forms in a document. It also allows the submission of forms to be protected by different access controls than the serving of forms. See 3.4. _ACCESS Files, page 22).

The EmWeb/Compiler generates the following function and structure prototypes:

```
typedef struct EwaForm_foo_s
 {
   struct
     {
       ...
     } value;
```

Copyright© 1997 Agranat Systems, Inc.

```
    struct
      {
        ...
      } status;
  } EwaForm_foo;...

void ewaFormServe_foo (EwsContext context, EwaForm_foo *formp);
char * ewaFormSubmit_foo (EwsContext context, EwaForm_foo *formp);
```

The ellipsis (...) above are replaced by declarations of the structure members to hold the data for each form element. The standard HTML NAME attribute is used in each INPUT tag of the form to specify the structure member names within the form data structure.

> *Restriction: While HTML allows arbitrary string values for NAME attributes, EmWeb requires that NAME attributes are valid C identifiers since they are used as field names in the form's C structure definition.*

The application-provided `ewaFormServe_foo` function is invoked by EmWeb/Server when the HTML document containing the form is requested by a Web browser. This provides the application with an opportunity to modify the default values displayed by the browser at run-time.

The application-provided `ewaFormSubmit_foo` function is invoked by EmWeb/Server when the corresponding HTML form is submitted. This provides the application with an opportunity to process the submitted form data and generate an appropriate response.

3.2.5.2. Form Data Structure

The generated form data structure passes information for the form between the system and the EmWeb/Server. It has two sections: 'value' and 'status', each containing one or more members for each form element. The 'status' section member is a flag that indicates the status of the contents of the 'value' member. The 'value' member contains the data for the form element in an internal representation which may be specified by the form author using the EMWEB_TYPE attribute. For example:

```
<INPUT TYPE=TEXT SIZE=15 MAXSIZE=15 VALUE=127.0.0.1
 NAME=ifAddr EMWEB_TYPE=DOTTED_IP>
```

The example above produces the following structure members:

```
typedef struct EwaForm_foo_s
 {
   struct
     {
       ...
       uint32     ifAddr;
```

```
        uintf       ifAddr_size;
        uintf       ifAddr_maxlength;
        ...
      } value;
    struct
      {
        ...
        uint8       ifAddr;
        ...
      } status;
  } EwaForm_foo;
```

The `value.ifAddr` member contains the host-order 32-bit IP address. The `ewaFormServe_foo` function sets this structure member with the value to be displayed on the form and sets `status.ifAddr` to `EW_FORM_INITIALIZED`. (Note that if `status.ifAddr` is not changed by the application, then the form is displayed with the value specified by the `VALUE` tag, "`127.0.0.1`"). When the Web browser end-user has entered a valid IP address and submits the form, the EmWeb/Server sets `value.ifAddr` to the value provided by the browser and sets `status.ifAddr` to `EW_FORM_RETURNED` before invoking `ewaFormSubmit_foo`.

The `value.ifAddr_size` member is generated because the `SIZE` attribute was specified and defaults to 15. This gives `ewaFormServe_foo` an opportunity to override the field size value before serving the form. (This field is not used by `ewaFormSubmit_foo`).

The `value.ifAddr_maxlength` member is generated because the `MAXSIZE` attribute was specified and defaults to 15. This gives `ewaFormServe_foo` an opportunity to override the maximum field size value at run-time. (This field is not used by `ewaFormSubmit_foo`).

The `status.ifAddr` member is used to indicate the status of the form element member and consists of a bit field defined as follows (from `src/include/ews_def.h`):

```
/*
 * For ewaFormServe_*
 */
#define EW_FORM_INITIALIZED 0x01    /* set if field initialized */
#define EW_FORM_DYNAMIC     0x02    /* set if field ewaAlloc'ed */

 /*
  * For ewaFormSubmit_*
  */
#define EW_FORM_RETURNED    0x10    /* set if value returned */
#define EW_FORM_PARSE_ERROR 0x20    /* set if value invalid */
#define EW_FORM_FILE_ERROR 0x20     /*        or i/o error */
```

For the `ewaFormServe_*` call (when the form is being sent to the browser), the structure is passed with the status for each element set to either zero, or `EW_FORM_INITIALIZED` if a default value was specified by the HTML `VAULE` tag. When

Copyright © 1997 Agranat Systems, Inc.

control is returned to the EmWeb/Server, the server checks each status and uses the new value for each field for which the EW_FORM_INITIALIZED is set. Initialized values override any VALUE, CHECKED, or SELECTED values specified in the source HTML document. The EW_FORM_DYNAMIC flag is only valid for type fields containing pointers to data (including TEXT, PASSWORD, HIDDEN, TEXTAREA, SUBMIT, and EMWEB_TYPE=HEXSTRING) and indicates that the memory referenced by the pointer was allocated by ewaAlloc and must be freed by EmWeb/Server using ewaFree after the form is served.

When the form is submitted by the browser, the server attempts to convert each value returned by the browser to the appropriate type. The server sets the status member to indicate the status of the data member of the structure, and then calls the ewaFormSubmit_* routine associated with the form. The EW_FORM_RETURNED flag is set if a valid value was returned for the field in the submission. The EW_FORM_PARSE_ERROR flag is set if a value was returned, but was not parsed correctly for the specified EMWEB_TYPE.

> *Note: When a form is submitted, the contents of a value field is undefined unless the EW_FORM_RETURNED flag is set in the corresponding status byte.*

The application processes the submitted values and returns a string containing the URL to use as the redirection response to the Web browser, or NULL for an empty ("204 No Content") response. Alternatively, the following function may be invoked to return a local relative URL from the archive as a response.

```
EwsStatus ewsContextSendReply ( EwsContext context, char * url );
```

3.2.5.3. HTML Form TEXT Input Fields

A text field can be used to represent a wide variety of value types. For example, a text field may be used to configure an IP address in which case the natural C-language representation is a host-order 32-bit integer. By default, a text field is simply a null-terminated character string. However, the EMWEB_TYPE attribute may be used to indicate any of a number of supported EmWeb types. Refer to the table below for details.

HTML text fields may specify a field size and/or a maximum length. If so, additional _size and _maxlength structure members are added to the form structure to give the application run-time control over these values. The format and resulting form structure fields of a text input field is given below.

```
<INPUT TYPE=TEXT NAME=name [MAXLENGTH=#max] [SIZE=#size] [VALUE=value]
 [EMWEB_TYPE=ewtype]>

EwType   name;            /* defaults to value */
uintf    name_maxlength;  /* defaults to #max */
uintf    name_size;       /* defaults to #size */
```

Copyright © 1997 Agranat Systems, Inc.

The following table lists all EMWEB_TYPE values currently supported.

EMWEB_TYPE	C-Type	Description
(not specified)	char *	null-terminated string
DECIMAL_INT	int32	32-bit signed integer
DECIMAL_UINT	uint32	32-bit unsigned integer
HEX_INT	uint32	32-bit unsigned hex integer
DOTTED_IP	uint32	32-bit host-order IP address translated using the conventional dotted-quad ASCII notation.
IEEE_MAC	char[6]	IEEE MAC address (any of ':' or '-' or '' is accepted between bytes on input, displayed with ':'). I/G bit is least significant (Ethernet)
FDDI_MAC	char[6]	FDDI MAC address (any of ':' or '-' or '' is accepted between bytes on input, displayed with '-'). I/G bit is most significant (FDDI, Token Ring)
STD_MAC	char[6]	A MAC address - if ':' or '' is used between bytes on input, it is interpreted as IEEE (I/G bit is least significant); if '-' is used on input, it is interpreted as FDDI (I/G bit is most significant). Displayed as IEEE with ':'.
HEX_STRING	struct { uint32 length; uint8 *datap; };	A hex string. Any of ':' or '-' or '' or white-space is accepted between bytes. Displayed with ''.
DECNET_IV_ADDRESS	uint16	area.node
OBJECT_ID	struct { uint32 length; uint32 *datap; };	A variable length array of unsigned decimal integers displayed with periods between elements.

Note that support for additional EmWeb types may be added in future releases.

3.2.5.4. HTML Form PASSWORD Input Fields

A password field is represented by a null-terminated character string. HTML password fields may specify a field size and/or a maximum length. If so, additional _size and _maxlength structure members are added to the form structure to give the application run-time control over these values. The format and resulting form structure fields of a password input field is given below.

Copyright © 1997 Agranat Systems, Inc.

```
<INPUT TYPE=PASSWORD NAME=name {SIZE=#size} {MAXLENGTH=#maxlength}
 {VALUE=value} >

char     *name;            /* defaults to value, or NULL*/
uintf    name_size;        /* defaults to #size */
uintf    name_maxlength;   /* defaults to #maxlength */
```

3.2.5.5. HTML Form CHECKBOX Input Fields

Checkbox fields are represented as booleans. The optional value field is passed unchanged to the browser and is not used by EmWeb.

```
<INPUT TYPE=CHECKBOX NAME=name {VALUE=value} {CHECKED}>

boolean  name;    /* defaults to TRUE if CHECKED, else FALSE */
```

Note: If the EW_FORM_RETURNED bit in the status byte is set on submission, then the checkbox was checked and the value is TRUE.

3.2.5.6. HTML Form RADIO Input Fields

Radio buttons are represented by a C enumerated type generated by the EmWeb/Compiler for each unique VALUE assigned to a radio button in the archive. (The enumerated types defined by radio buttons and single select options are combined into a single archive-wide enumeration).

```
<INPUT TYPE=RADIO NAME=name VALUE=value1 {CHECKED}>
<INPUT TYPE=RADIO NAME=name VALUE=value2 {CHECKED}>

typedef enum EwaFormEnum_ew_archive_e
 {
   ...
  ,value1
  ,value2
   ...
 } EwaFormEnum_ew_archive;

EwaFormEnum_ew_archive name;/* defaults to checked value */
```

Restriction: While HTML allows arbitrary string values for VALUE attributes, EmWeb requires that VALUE attributes for radio buttons are valid C identifiers since they are used as enumerated type identifiers.

Note that EmWeb/Compiler generates a single enumeration for each archive. The *ew_archive* suffix may be overridden at the command line (See 3.5. Compiler Options, page 23).

The following function is provided as a convenience for translating from the

Copyright © 1997 Agranat Systems, Inc.

enumerated type's integer value to its name as specified by the VALUE attribute:

```
const char *ewsFormEnumToString ( EwsContext context, int value );
```

3.2.5.7. HTML Form IMAGE Input Fields

Images are "read only" from the point of view of the application as they return the x and y image coordinates selected by the end-user from the Web browser. Therefore, the generated structure value members are only useful in the ewaFormSubmit_* application function. Structure members are generated representing the x and y coordinates as follows.

```
<INPUT TYPE=IMAGE NAME=name SRC=src {ALIGN=align}>

typedef struct EwaForm_foo_s
{
  struct
    {
      ...
      uint32  name_x;
      uint32  name_y;
      ...
    } value;
  struct
    {
      ...
      uint8   name;
      ...
    } status;
} EwaForm_foo;
```

3.2.5.8. HTML Form HIDDEN Input Fields

Hidden fields are identical to text fields except that they are not displayed (and thus not modified) by the browser. They are typically used by applications to pass context information to the browse. This context information is returned to the application during submission.

```
<INPUT TYPE=HIDDEN NAME=name {VALUE=value} {EMWEB_TYPE=ewtype} >

EwType name;        /* defaults to value if present */
```

3.2.5.9. HTML Form SUBMIT Input Fields

Submit buttons with NAME attributes are represented by a null-terminated character string. Submit buttons without NAME attributes are not included in the form structure.

```
<INPUT TYPE=SUBMIT {NAME=name} {VALUE=value} >

char *name;         /* defaults to value if present */
```

Copyright © 1997 Agranat Systems, Inc.

3.2.5.10. HTML Form RESET Input Fields

No structure member is generated for a reset button.

3.2.5.11. HTML Form SELECT/OPTION Fields

Select boxes may be used to select a single item from a list (the default), or multiple items from a list (if the attribute MULTIPLE is present). For single item selection, values are represented by a C enumerated type generated by the EmWeb/Compiler. (The enumerated types defined by radio buttons and single select options are combined into a single archive-wide enumeration).

```
<SELECT NAME=name {SIZE=#size} >
        <OPTION VALUE=value1 {SELECTED} >
        <OPTION VALUE=value2 {SELECTED} >
</SELECT>

typedef enum EwaFormEnum_ew_archive_e
 {
   ...
  ,value1
  ,value2
   ...
 } EwaFormEnum_ew_archive;

EwaFormEnum_ew_archive name;/* defaults to selected value */
```

For multiple item selection, a status and value form structure field is generated for each option as follows:

```
<SELECT NAME=name {SIZE=#size} MULTIPLE >
        <OPTION VALUE=value1 {SELECTED} > descriptive text
        <OPTION VALUE=value2 {SELECTED} > descriptive text
</SELECT>

boolean value1;          /* defaults to TRUE if SELECTED */
boolean value2;          /* defaults to TRUE if SELECTED */
uintf name_size;         /* defaults to #size */
```

HTML select fields may specify the size of the selection box. If so, an additional _size structure member is added to the form structure to give the application run-time control over this value.

> ***Restriction:*** *While HTML allows arbitrary string values for* VALUE *attributes, EmWeb requires that* VALUE *attributes for select options are valid C identifiers since they are used as enumerated type or structure field identifiers.*

3.2.5.12. TEXTAREA

The textarea value is a null-terminated string of characters. Additional value fields are placed in the form structure for the number of rows and columns in the textarea.

Copyright © 1997 Agranat Systems, Inc.

These fields are initialized by EmWeb/Server to the defaults specified in the source HTML document, but may be overridden at run-time by the application's `ewaFormServe_*` function.

```
<TEXTAREA COLS=#cols ROWS=#rows NAME=name >
...
</TEXTAREA>

char     *name;          /* null-terminated string */
uintf    name_cols;      /* default to #cols */
uintf    name_rows;      /* default to #rows */
```

Note that the `*name` points to the text present in the `TEXTAREA` input field.

3.2.5.13. File Upload

RFC1867 file upload defines a new form input type as follows:

```
<INPUT TYPE=FILE NAME=name {VALUE=value} {SIZE=#rows{,#cols}}>
```

> Note: Use of file input types requires that the form uses `multipart/form-data` as an encapsulation type. This must be specified in the `<FORM>` tag as follows:
>
> ```
> <FORM METHOD=POST ENCTYPE=multipart/form-data ... >
> ```

This causes the browser to prompt the user for a local (relative to the browser) filename (by default, value is used if specified). On submission, the file is uploaded to the server along with other form field values.

EmWeb/Compiler parses file type input fields into the following fields in the form value substructure:

```
char             *name;          /* serve only: default filename */
EwaFileHandle    name_handle;    /* submit only: received file */
uintf            name_size;      /* serve only: #rows (no #cols) */
uintf            name_rows;      /* serve only: #rows (w/#cols) */
uintf            name_cols;      /* serve only: #cols */
```

In addition, the following fields in the form status substructure are generated as well:

```
uint8            name;           /* status for filename */
uint8            name_handle;    /* status for received file */
```

For more detailed information on the file upload interface, please refer to 4.1.8. Local Filesystem Interfaces, page 54.

3.2.6. Graphic Maps

The HTML `ISMAP` attribute on an `IMG` tag specifies that the file is an image map. Such an `IMG` tag must always be enclosed within an anchor that specifies a hyper-link URL.

Copyright © 1997 Agranat Systems, Inc.

The ISMAP attribute causes a click within the image to be sent as a request to the URL in the enclosing anchor, qualified by the x & y coordinates of the click within the image.

EmWeb provides support for image maps through a map element in the HTML source tree which specifies rectangular regions and URLs to be associated with them.

The interface designer creates an image map file, a text file which defines the URLs to be provided for each region of the image. The syntax of the entries is:

```
rect-line | default-line

rect-line          ::= rect url point point

point              ::= x,y

default-line       ::= default url
```

The `rect` token specifies a URL to be served in response to the specified top-left and bottom-right coordinates. The optional `default` specifier provides the URL for any coordinate not specified by some `rect` (only one `default` is allowed). Order of the `rect` lines is significant - if the regions overlap the first match takes precedence.

If no match is found, and no default is specified, the server returns "no content" to the browser, effectively a no-operation. Otherwise, the matching URL is returned to the browser as a redirection. The URL may be relative to the map file in the archive, or absolute.

The map files are compiled by EmWeb/Compiler so that they do not need to be parsed at run-time by the EmWeb/Server. The following illustrates the syntax of a typical map file:

```
rect example1.html 15,18 169,37
rect example2.html 157,46 385,65
rect example3.html 365,71 460,87
default /home.html
```

If the above file was "`sample.map`", and corresponded to the image "`/pictures/sample.gif`", the following HTML would be used to implement the map:

```
<A HREF="sample.map"> <IMG SRC="/pictures/sample.gif" ISMAP> </A>
```

By convention, map files use the suffix '`.map`', but this can be changed. See 3.3. mime.types Configuration File, page 20, and 3.4. _ACCESS Files, page 22.

3.2.7. Cache Control

The HTTP/1.0 protocol has some simple provisions for communicating cachability information to caches and proxies between the server and the browser. HTTP/1.1 provides even greater cache control features.

Copyright © 1997 Agranat Systems, Inc.

By default, EmWeb/Server classifies compiled-in documents as being either static or dynamic. A document is considered dynamic if it contains one or more EMWEB_STRINGS, EMWEB_INCLUDES, or EmWeb-enhanced HTML forms. Otherwise, the document is static.

When serving a static document, EmWeb/Server sends a Last-Modified HTTP header indicating the time and date that the corresponding archive was created by the EmWeb/Compiler. No other cache control headers are generated in HTTP/1.0. With HTTP/1.1, a Cache-Control header is generated indicating either "public" if the document is not protected by an authentication realm, or "private" if it is.

When serving a dynamic document, EmWeb/Server sends a Last-Modified and Expires HTTP header indicating the current time and date, as well as "Pragma: no-cache" for HTTP/1.0 and "Cache-Control: no-cache" for HTTP/1.1.

It may be desirable to override the above default behavior for certain applications. Specifically, an application may desire static treatment of a dynamic document. For example, if EMWEB_INCLUDE tags are used to insert common headers and footers, EmWeb/Server would treat the document as dynamic even though the content may actually be static. The following EmWeb tag may be used anywhere in an HTML document to mark it as static for cache control purposes:

```
<EMWEB_STATIC>
```

3.2.8. EmWeb CGI

The mechanisms described above are designed to meet most needs of the system designer without having to deal with the complexities of direct access to the HTTP input and output mechanisms. If such access is desired, the EmWeb raw CGI interface provides access similar to that available in a CGI program environment on any HTTP server. See 4.1.6. Raw CGI, page 51.

3.3. mime.types Configuration File

The `mime.types` configuration file is read by EmWeb/Compiler before processing source directories of HTML and other Web documents. If "-m" is specified on the EmWeb/Compiler command line, then the specified file is used. Otherwise, EmWeb/Compiler searches for a `mime.types` file by first looking at the $EMWEB_MIME environment variable, and then looking at $EMWEB_HOME/$EMWEB_MIME. If the $EMWEB_MIME environment variable is not specified, it defaults to "mime.types". If the $EMWEB_HOME environment variable is not specified, it defaults to "/usr/local/share/emweb".

The `mime.types` file contains default parameters for files derived from the file name suffix. For example:

```
#
# EmWeb/Compiler mime.types file example
# Anything after a '#' is a comment.
# A line whose first character is white space is a continuation line
```

Copyright © 1997 Agranat Systems, Inc.

```
# Each specifier must end in ';'
#

.html    type=text/html parse=emweb_html compress ;
.txt     type=text/plain parse=emweb_text compress ;
.gif     type=image/gif parse=binary ;
.map     imagemap;
.        index=index.html;
```

The suffix may be any tail match; it is not restricted to values starting with '.'. The `mime.types` file contains specifiers for the default access rights and parameters of files ending with a particular suffix. Each specifier is the file name suffix, followed by one or more of the following attributes, followed by a semicolon:

type=*mime-type*

Specifies the MIME encapsulation media type represented by the file. Media type values are registered with the Internet Assigned Number Authority (IANA). Use of non-registered media types is discouraged. (Note that the *mime-type* must be quoted if it contains a ';' line-terminator character). The default type depends on the parser selected (see below).

parse=*emweb_html | emweb_text | text | binary*

Specifies how the EmWeb/Compiler is to parse the file.

EMWEB_HTML indicates that the file contains HTML and the EmWeb/Compiler handles <EMWEB_STRING>, <EMWEB_INCLUDE>, <EMWEB_PROTO>, <EMWEB_HEAD>, and EmWeb HTML forms. The default type for the EMWEB_HTML parser is "text/html".

EMWEB_TEXT indicates that the file contains text which may contain <EMWEB_STRING>, <EMWEB_INCLUDE>, <EMWEB_PROTO> and <EMWEB_HEAD> directives. However, such a file may not contain EmWeb HTML forms. The default type for the EMWEB_TEXT parser is "text/plain".

TEXT indicates that the file contains text which should not be parsed (e.g. PostScript). The default type for the TEXT parser is "text/plain".

BINARY indicates that the file contains raw binary data. The default type for the BINARY parser is "application/octet-stream".

The default parser is BINARY if not otherwise specified.

compress

Specifies that the content should be compressed. The EmWeb compression algorithm creates an archive-wide dictionary of common strings which are referenced by the individual documents. Compression ratios for text files of 50% can be achieved if there is sufficient redundancy

Copyright © 1997 Agranat Systems, Inc.

throughout the archive. However, compression ratios will vary widely among applications. The default is no compression.

nocompress
Specifies that the content should not be compressed. (This is typically used in an _ACCESS file to override a specification in `mime.types`). The default is no compression.

imagemap
Specifies that the content is an imagemap that defines the coordinates for an image. This implies "hidden", and inherits realm membership from the current directory.

ignore
Specifies that the entry should be skipped (i.e. not included in the archive). This is typically used to exclude source control directories, backup files from editors, etc.

The current directory is specified using '.' as the suffix name. Directories may contain the attribute:

index=*index-file-name*
The element (in the directory) to be returned in response to an access request to that directory (a URL whose last component is the directory name).

3.4. _ACCESS Files

Access to individual files and directories is controlled by a configuration file in each source archive directory named '_ACCESS'. The _ACCESS files contain specifiers for the access rights of elements in the corresponding directory (and optionally for the directory itself using the '.' specifier). Specifiers in the _ACCESS file override specifiers in the `mime.types` file. Each specifier is the file name (not a suffix) followed by any of the attributes defined above for `mime.types`. The following additional attributes are permitted in _ACCESS files.

realm=*realm-name*
requires that any request to access the file or directory must be authenticated in the named realm. If an empty realm is given (i.e. `realm=""`), then the file may be accessed without any authentication. See 4.1.4. Authentication and Security, page 39.

hidden
Specifies that the file is included for use in constructing dynamic content and may not be seen or accessed directly by a Web browser. (Note that the hidden and realm attributes are mutually exclusive).

Copyright © 1997 Agranat Systems, Inc.

link=*url*

Specifies that this element is linked to a different URL. When accessing this file, the browser shall be redirected to the specified relative or absolute URL. The link attribute may not appear with any other attributes.

If `realm` is specified for a directory, it applies as a default for all files and subdirectories within the current directory. Specifications for individual files in an `_ACCESS` file override defaults specified for the directory.

If `hidden` is specified for a directory, it applies as a default for all files and subdirectories within the current directory. Individual files and subdirectories may be overridden by specifying a `realm`.

Note that a default '.' specification for index may be included in the `mime.types` file and overridden in individual `_ACCESS` files.

In addition to the elements in the directory, the `_ACCESS` file may specify the base name for CGI elements (which do not need actual files in the HTML tree). These are specified with the access attribute cgi below and may also have a `realm` attribute.

cgi=*symbol*

The element is a CGI application. The symbol value is used by the EmWeb/Compiler to generate application-specific prototypes for the raw-CGI interface. See 4.1.6. Raw CGI, page 51.

Finally, form action URLs used when a browser submits an EmWeb HTML form to the EmWeb/Server may be protected by a specific realm which may be different from the realm of the document containing the form. In this way a form may be used to display data to a less restricted realm, while permitting changes only from users in a more restricted realm. For example, if the HTML file "`example.html`" contained an EmWeb HTML form tag `<FORM METHOD=POST EMWEB_NAME=foo ACTION=bar>`, the following specifier may appear in the `_ACCESS` file corresponding to the directory containing "`example.html`".

```
example.html/bar  realm=realm-name
```

Access control is only computed by the EmWeb/Server on the initial URL; if the element to be returned in response to the URL includes an `EMWEB_INCLUDE` tag, the EmWeb/Server does not perform an access check on those individual elements.

3.5. Compiler Options

The EmWeb/Compiler program "ewc" takes the following command-line arguments followed by a list of one or more directories and/or files with which to build an EmWeb archive.

Copyright © 1997 Agranat Systems, Inc.

(-n | --name) <archive-symbol-base-name>

The base symbol name produced by EmWeb/Compiler corresponding to the object part (and, if -c is also specified, the data part) of the archive. By default, this parameter is "`ew_archive`". Thus, the following symbol is generated by the EmWeb/Compiler in `ew_code.c`:

```
EwsArchive ew_archive;
```

And if the -c option is specified, the following symbol is generated in `ew_data.c`.

```
uint8 ew_archive_data[];
```

(-m | --mime) <mime.types filename>

The path name to the `mime.types` configuration file. By default, EmWeb/Compiler looks in `$EMWEB_MIME` (*`mime.types`*) followed by `$EMWEB_HOME/$EMWEB_MIME` (*`/usr/local/share/emweb/mime.types`*).

(-u | --url) <archive-url-base-path>

The base URL path of the generated archive. By default, this is '/'.

(-r | --raw) | (-c | --c)

The output format of the generated archive data component (raw by default). If `--raw` is specified, the data component of the archive is written to the file `ew_data.dat` as a raw binary file. If `--c` is specified, the data component of the archive is written to the file `ew_data.c` as a C file that, when compiled, produces an array of data reference by the symbol `ew_archive_data`. Both flags may be specified on the command line to cause the generation of both raw and C output files.

(-o | --output)

The prefix for generated files `data.dat`, `data.c`, `code.c`, `stubs.c`, and `proto.h` ("`./ew_`" by default).

(-s | --stubs)

Generate `ew_stubs.c` file containing stubbed-out versions of form and CGI functions normally provided by the application to facilitate rapid prototyping and development.

(-C | --no-compress)

Disable compression of the archive. (Overrides `compress` attribute in `mime.types` and `_ACCESS` files).

(-l | --little)

The byte-order of the generated archive data component. By default, a big-endian archive is generated. If `--little` is specified, then a little-endian

Copyright © 1997 Agranat Systems, Inc.

archive is generated.

(-P | --no-preprocessor)
Disable generation of `#line` pre-processor directives in generated `ew_code.c` file; normally, these directives are included so that the C compiler and debuggers can find the original C in the source HTML documents. This option should only be used if your C compiler does not understand the '`#line`' directive.

(-q | --quiet) | (-v | --verbose) | (-d | --debug)
The logging level. The --quiet options suppresses all but error messages. The `--verbose` is specified, more logging output is produced. The `--debug` is specified, still more logging output is produced. The default (non of these options specified) includes errors and some warning messages.

(-v | --version)
Displays the EmWeb/Compiler's version.

--
Two dashes in a row indicate the end of options. All remaining parameters are file and/or directory names. This is optional, as any parameter that does not begin with a '-' is assumed to be a file and/or directory name.

3.6. Compiler Output

The EmWeb/Compiler produces the following output files:

ew_data.dat
A raw binary file containing the data component of the archive.

ew_data.c
If the `--c` option is specified at the command line, this file is generated. It contains C code that, when compiled, defines an array of octets representing the data component of the archive and referenced by the global symbol `uint8 ew_archive_data[]`.

ew_code.c
A C file that, when compiled, defines the object component of the archive referenced by the global symbol `(EwsArchive) ew_archive`.

ew_proto.h
This C header file is generated if the archive contains EmWeb HTML forms or CGI documents. The header file defines function prototypes and data structures for an application-specific interface between EmWeb/Server and application-provided functions responsible for serving and submitting forms and/or raw CGI processing.

Copyright © 1997 Agranat Systems, Inc.

ew_stubs.c

If the `--stubs` option is specified at the command line, this file is generated. It contains stubbed-out versions of form and CGI functions normally provided by the application to facilitate rapid prototyping and development.

Copyright © 1997 Agranat Systems, Inc.

4. EmWeb/Server

The EmWeb/Server is written in portable ANSI C and is easily integrated into a wide variety of embedded application environments. EmWeb/Server implements the HTTP protocol and responds to requests from networked Web browsers with documents stored in run-time archives created by the EmWeb/Compiler.

The following figure illustrates a typical EmWeb/Server device management application.

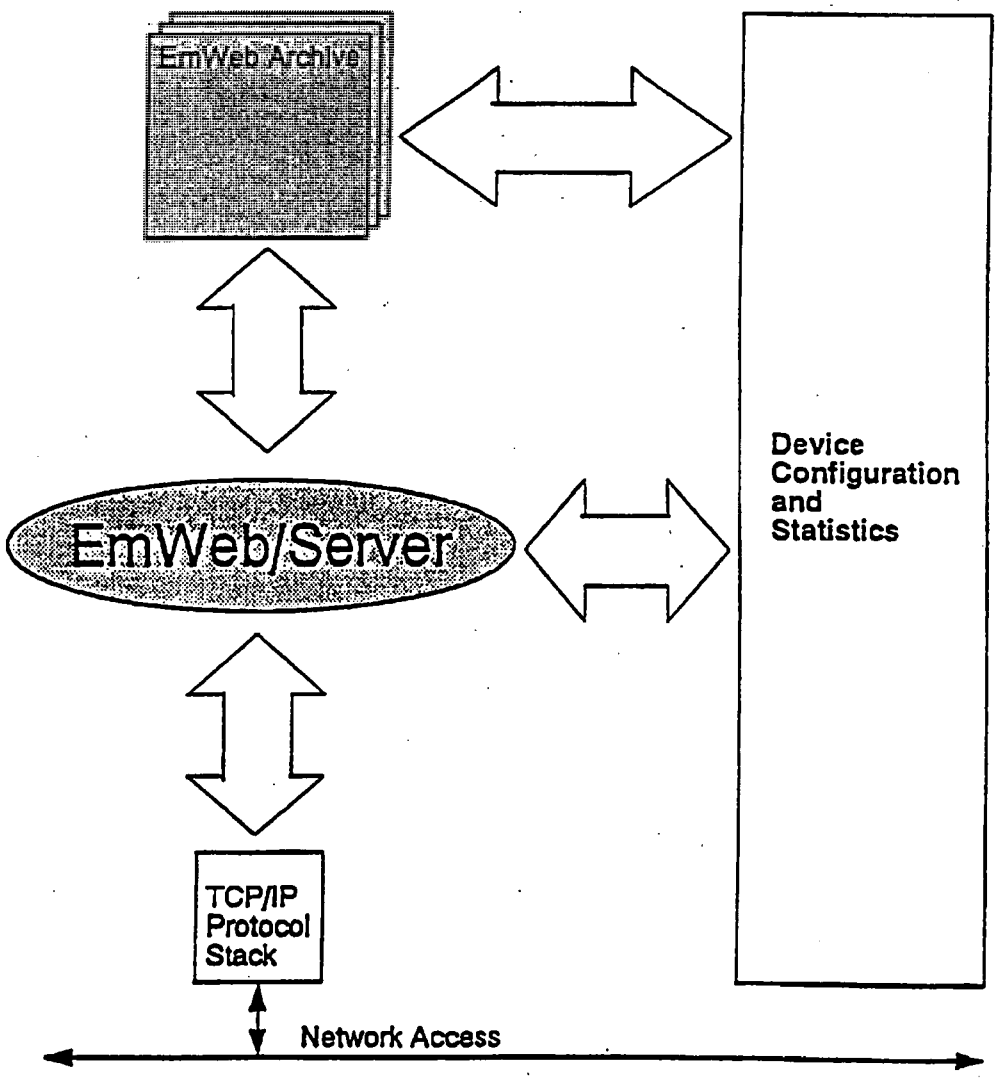


Figure 2: EmWeb/Server

4.1. Application Interfaces

The EmWeb/Server application interfaces are divided into six functional areas including system interfaces, network interfaces, document and archive maintenance, authorization and security, request context access, and raw CGI.

Many application interfaces are optional and can be configured out of the EmWeb/Server at compile time. This offers the system integrator flexibility in balancing the trade-offs between functionality, memory requirements, and system performance.

Most application interfaces are provided by the EmWeb/Server and are called by the application. However, some application interfaces must be provided by the application and are called by the EmWeb/Server. Function names beginning with "ews" and data types beginning with "Ews" are defined by the server for use by the application. Function names beginning with "ewa" and data types beginning with "Ewa" must be defined by the application for use by EmWeb/Server.

Status codes returned by the EmWeb/Server to the application are of type EwsStatus and are defined as follows (from `src/include/ews_def.h`):

```
/*
 * Status codes returned to the application by EmWeb/Server
 */
typedef enum EwsStatus_e
  {
    EWS_STATUS_OK,
    EWS_STATUS_BAD_MAGIC,
    EWS_STATUS_BAD_VERSION,
    EWS_STATUS_ALREADY_EXISTS,
    EWS_STATUS_NO_RESOURCES,
    EWS_STATUS_IN_USE,
    EWS_STATUS_NOT_REGISTERED,
    EWS_STATUS_NOT_CLONED,
    EWS_STATUS_NOT_FOUND,
    EWS_STATUS_AUTH_FAILED,
    EWS_STATUS_BAD_STATE,
    EWS_STATUS_BAD_REALM,
    EWS_STATUS_FATAL_ERROR,
    EWS_STATUS_ABORTED
  } EwsStatus;
```

Status codes returned by the application to the EmWeb/Server are of type EwaStatus and are defined as follows (from `src/include/ews_def.h`):

```
/*
 * Status codes returned to EmWeb/Server by the application
 */
typedef enum EwaStatus_e
  {
    EWA_STATUS_OK,
    EWA_STATUS_OK_YIELD,
    EWA_STATUS_ERROR
```

Copyright © 1997 Agranat Systems, Inc.

```
} EwaStatus;
```

4.1.1. System Interfaces

The system functional interfaces are further divided into four functional areas including initialization, scheduling, memory management, and time-of-day services.

4.1.1.1. Initialization and Shutdown

Before any other application interface function may be used, the EmWeb/Server must be initialized. This is accomplished by invoking the following function:

```
EwsStatus ewsInit ( void );
```

This function returns EWS_STATUS_OK on success. Otherwise, an error code is returned. (EWS_STATUS_NO_RESOURCES is returned if EmWeb/Server was unable to allocate memory for internal hash tables, etc.).

A graceful shutdown of the EmWeb/Server can be accomplished by invoking the function below. This causes EmWeb/Server to terminate all outstanding HTTP requests and release all dynamically allocated resources.

```
EwsStatus ewsShutdown ( void );
```

This function returns EWS_STATUS_OK on success. Otherwise, an error code is returned. (There are currently no conditions that cause an error to be returned).

4.1.1.2. Scheduling

The EmWeb/Server is capable of serving multiple HTTP requests simultaneously. It may be configured at compile-time to use either its own internal real-time scheduler or to make use of an external scheduler provided by the application's operating system.

If an external scheduler is to be used, HTTP requests may be multi-threaded if each TCP connection is handled by its own operating system thread. Otherwise, requests will be single-threaded. The `ewsRun()`, `ewsSuspend()` and `ewsResume()` functions described below for EmWeb/Server's internal scheduler would not be used. Instead, all application <EMWEB_STRING> and <EMWEB_INCLUDE> code fragments and `ewaFormServe_*` and `ewaFormSubmit_*` functions are invoked from the `ewsNetHTTPReceive()` function.

If the internal scheduler is to be used, HTTP requests are served from the following function:

```
EwsStatus ewsRun ( void );
```

This function returns EWS_STATUS_OK on success. Otherwise, an error code is returned. (There are currently no conditions that cause an error to be returned). All application <EMWEB_STRING> and <EMWEB_INCLUDE> code fragments and

Copyright © 1997 Agranat Systems, Inc.

ewaFormServe_* and ewaFormSubmit_* functions are invoked from ewsRun().

The ewsRun() function processes any outstanding HTTP requests until either all requests have been served, or the application requests that EmWeb/Server yield the CPU as follows. Each time EmWeb/Server sends a buffer of HTTP response data to the network via the application-provided ewaNetHTTPSend function, the application may return the status EWA_STATUS_OK_YIELD to request that the EmWeb/Server relinquish control of the CPU. This will cause the EmWeb/Server to return from the ewsRun() function immediately. (Note that if the last buffer of the HTTP response was transmitted, the EmWeb/Server will close the connection and release any corresponding resources before returning control to the application).

In addition, the internal scheduler gives the application a means to temporarily suspend a specific HTTP request each time it invokes application-specific code from <EMWEB_STRING>, <EMWEB_INCLUDE>, and HTML form serve and submit functions. The application may instruct EmWeb/Server to suspend an HTTP request from the application-specific code provided in <EMWEB_STRING> and <EMWEB_INCLUDE> C code fragments, and EwaFormServe_* and EwaFormSubmit_* functions by invoking the following function before returning to the EmWeb/Server.

```
EwsStatus ewsSuspend ( EwsContext context );
```

This function returns EWS_STATUS_OK on success. Otherwise, an error code is returned (EWS_STATUS_BAD_STATE). When the application returns to EmWeb/Server after invoking ewsSuspend, the EmWeb/Server may continue to process other outstanding requests from ewsRun() before returning control to the application. Note that the value returned by the application after calling ewssuspend is ignored. The application's <EMWEB_STRING> or <EMWEB_INCLUDE> C code fragment or ewaFormServe_* or ewaFormSubmit_* function is re-invoked when the request is resumed. This functionality makes it possible to implement proxy servers (i.e. a local HTTP request might cause the application to send an IPC message to another process and suspend EmWeb/Server processing of this request until an IPC response is received). The application can resume a previously suspended request by invoking the following function.

```
EwsStatus ewsResume ( EwsContext context, EwaStatus status );
```

Status can be either EWA_STATUS_OK or EWA_STATUS_OK_YIELD. If the status is EWA_STATUS_OK, then the resumed request may be scheduled right away (for example, this could cause ewsRun to be invoked internally). Otherwise, the suspended request is scheduled to run, but is not actually run until control is transferred back to EmWeb/Server. This gives the application control over which CPU thread processes HTTP requests (i.e. a suspended request may be resumed by an interrupt handler for execution later in a polling loop that calls ewsRun). This function returns EWS_STATUS_OK on success. Otherwise, an error code is returned (EWS_STATUS_BAD_STATE).

Copyright © 1997 Agranat Systems, Inc.

The following function may be invoked from within the application's <EMWEB_STRING> or <EMWEB_INCLUDE> C code fragment or ewaFormServe_* or ewaFormSubmit_* function to determine if this is the initial invocation of the application code, or if the code is being resumed.

```
boolean ewsContextIsResuming ( EwsContext context );
```

This function returns TRUE if the application code is being re-invoked after ewsResume. Otherwise, this function returns FALSE.

4.1.1.3. Memory Management

The EmWeb/Server requires the application to provide simple memory management for the allocation and release of temporary data-structures used throughout the processing of HTTP requests and maintaining the run-time document archive and authentication databases. These functions are equivalent to the POSIX malloc and free calls as follows:

```
void * ewaAlloc ( uintf bytes );
void ewaFree ( void * pointer );
```

The ewaAlloc function (or macro) returns a pointer to available memory of at least the specified number of bytes long, or NULL if no memory is available. The ewaFree function (or macro) returns a block of available memory previously allocated by ewaAlloc.

> Note: If the EmWeb/Server is unable to allocate memory during the processing of a particular HTTP request, the request is aborted, all resources related to the request are released, and the application's ewaNetHTTPEnd() function is invoked. Alternatively, the application may implement ewaAlloc() so as to block until resources are available using external operating system facilities.

4.1.1.4. Time-of-Day Management

Some of the optional HTTP/1.0 functionality requires a "time-of-day" for Date:, Expires:, and Last-modified: headers. Many embedded systems do not have a reliable time-of-day capability, so these features are configurable options in EmWeb/Server. To take advantage of these features, the application must provide the following function (or macro):

```
const char * ewaDate ( void );
```

This function should return the current time (GMT) in one of the following two formats:

RFC1123: (Preferred)

```
Fri, 08 Jun 1996 08:57:28 GMT
```

Copyright© 1997 Agranat Systems, Inc.

RFC850:

```
Friday, 08-Jun-96 08:57:28 GMT
```

4.1.2. Network Interfaces

The EmWeb/Server relies upon a pre-existing TCP/IP protocol stack available in the target software environment. The interface between the application's TCP/IP protocol stack and the EmWeb/Server is event driven. This approach makes it possible to port EmWeb/Server to a wide variety of software environments.

4.1.2.1. Network Buffer Management

The EmWeb/Server design assumes that the application maintains data buffers for the reception and transmission of TCP data.

Buffers can be uniquely identified by an application-defined buffer descriptor. No assumptions are made about the actual structure of the buffer descriptors or their relationship to the data. For example, a buffer descriptor could be an index into a table, a pointer to a structure (either contiguous or separate from the data represented), etc. The application is responsible for defining the appropriate type for `EwaNetBuffer` and a value for `EWA_NET_BUFFER_NULL`.

Buffers can be chained together. Given a buffer descriptor, EmWeb/Server must be able to get or set the "next buffer in the chain" field. This is done by the `ewaNetBufferNextSet` and `ewaNetBufferNextGet` functions (or macros). Note that the buffer chain is terminated when the next buffer value is `EWA_NET_BUFFER_NULL`.

```
EwaNetBuffer ewaNetBufferNextGet ( EwaNetBuffer buffer );

void ewaNetBufferNextSet ( EwaNetBuffer buffer, EwaNetBuffer next );
```

Given a buffer descriptor, EmWeb/Server can determine the start of data in the buffer. Additionally, EmWeb/Server may change the start of data in the buffer (EmWeb/Server only moves the pointer to the data upward (increments)). This is done by the `ewaNetBufferDataGet` and `ewaNetBufferDataSet` functions (or macros).

```
uint8 * ewaNetBufferDataGet ( EwaNetBuffer buffer );
void ewaNetBufferDataSet ( EwaNetBuffer buffer, uint8 *datap );
```

Given a buffer descriptor, EmWeb/Server can determine the size of contiguous data in the buffer. Additionally, EmWeb/Server may change the size of the buffer (EmWeb/Server only changes the size of the buffer downward (decrements)). This is done by the `ewaNetBufferLengthGet` and `ewaNetBufferLengthSet` functions (or macros).

```
uintf ewaNetBufferLengthGet ( EwaNetBuffer buffer);
void ewaNetBufferLengthSet ( EwaNetBuffer buffer, uintf length );
```

Copyright © 1997 Agranat Systems, Inc.

EmWeb/Server may allocate a single buffer by invoking the `ewaNetBufferAlloc` function (or macro). The application may return a buffer of any size to EmWeb/Server. The size of the buffer must be initialized to the number of bytes available in the buffer for EmWeb/Server. If no buffers are available, this function returns `EWA_NET_BUFFER_NULL`. Additionally, EmWeb/Server may release a chain of one or more buffers by invoking the `ewaNetBufferFree` function (or macro).

```
EwaNetBuffer ewaNetBufferAlloc ( void );
void ewaNetBufferFree (EwaNetBuffer buffer);
```

Note: If the EmWeb/Server is unable to allocate a network buffer during the processing of a particular HTTP request, the request is aborted, all resources related to the request are released, and the application's `ewaNetHTTPEnd()` function is invoked. Alternatively, the application may implement `ewaNetBufferAlloc()` so as to block until resources are available using external operating system facilities.

4.1.2.2. TCP/IP Interfaces

The application is responsible for listening to the HTTP TCP port (80) for connection requests. When an HTTP TCP connection request is received, the application accepts the TCP connection on behalf of EmWeb/Server and invokes the following function:

```
EwsContext ewsNetHTTPStart ( EwaNetHandle handle );
```

This function returns a new context handle which the application is expected to use when referencing this HTTP request for future operations. The value `EWS_CONTEXT_NULL` is returned on failure (e.g. no resources available to handle the new request). The handle parameter is application-defined and is returned to the application unchanged by the EmWeb/Server in other network interface calls as illustrated below. The handle is also made available to `<EMWEB_STRING>` and `<EMWEB_INCLUDE>` C code fragments and `ewaFormServe_*` and `ewaFormSubmit_*` application functions by invoking the following function:

```
EwaNetHandle ewsContextNetHandle ( EwsContext context );
```

At any time, the application can abort an uncompleted HTTP request by invoking the following function:

```
EwsStatus ewsNetHTTPAbort ( EwsContext context );
```

This function is typically invoked by the application to notify EmWeb/Server that the corresponding TCP/IP connection has been disconnected. The function returns `EWS_STATUS_OK` on success. Otherwise, an error code is returned. (There are

Copyright © 1997 Agranat Systems, Inc.

currently no conditions that cause an error to be returned).

As data buffer(s) are received from the network on a TCP connection corresponding to an HTTP request, the application passes these buffers to the EmWeb/Server by invoking the following function.

```
EwsStatus ewsNetHTTPReceive (EwsContext context, EwaNetBuffer buffer);
```

This function returns EWS_STATUS_OK on success. Otherwise, an error code is returned. (There are currently no conditions that cause an error to be returned). The buffer parameter describes a chain of one or more buffers containing raw TCP data. The structure of the buffers is determined by the application, and the application provides EmWeb/Server with functions to manipulate these buffers as defined previously. Note that EmWeb/Server takes responsibility for releasing received buffers when they are no longer needed, and may hold on to buffers containing certain HTTP request headers for the duration of the request.

When EmWeb/Server has assembled one or more buffers for transmission, it invokes the following function (or macro) which must be provided by the application:

```
EwaStatus ewaNetHTTPSend ( EwaNetHandle handle, EwaNetBuffer buffer );
```

This function transmits (and then releases) a chain of one or more network buffers and returns EWA_STATUS_OK or EWA_STATUS_OK_YIELD on success, or EWA_STATUS_ERROR on failure. If the application returns EWA_STATUS_OK, EmWeb/Server will continue processing outstanding requests in a round-robin fashion. If the application returns EWA_STATUS_ERROR, EmWeb/Server will abort the current request (as if ewsNetHTTPAbort() was called) and continue processing any other outstanding requests. The application returns EWA_STATUS_OK_YIELD to request that EmWeb/Server give up the CPU. In most cases, ewaNetHTTPSend() is called for each buffer's worth of HTTP response data generated by the EmWeb/Server. Thus, the CPU used by the server can be throttled by adjusting the buffer size available to the server and using the EWA_STATUS_OK_YIELD return code. (See 4.1.1.2. Scheduling, page 29). Note that use of this return code requires that the application eventually either reschedule processing of the request by invoking ewsRun, or abort the request by invoking ewsNetHTTPAbort.

The application may signal the EmWeb/Server with flow control events to indicate congestion on an outbound TCP connection. If a TCP connection's transmit window is full, the application may call the following function at any time:

```
EwsStatus ewsNetFlowControl ( EwsContext context );
```

This will cause the EmWeb/Server to temporarily suspend processing of the request. When the TCP connection's window opens again, the application may invoke the following function at any time to notify EmWeb/Server that processing of the request may continue:

Copyright© 1997 Agranat Systems, Inc.

```
EwsStatus ewsNetUnFlowControl ( EwsContext context);
```

If `ewsNetFlowControl()` is called from the `ewaNetHTTPSend()` function, then the application is responsible for transmitting the current buffer. However, `ewaNetHTTPSend()` will not be called again until the application has called `ewsNetUnFlowControl()`. Note that if `ewsNetFlowControl()` is invoked from outside `ewaNetHTTPSend()`, then EmWeb/Server may call `ewaNetHTTPSend()` once before suspending the request. In any case, the application should be prepared to receive notification of request completion as described below.

When the EmWeb/Server has completed processing an HTTP request, or if the application aborts an outstanding HTTP request by invoking `ewsNetHTTPAbort()`, it invokes the following function which must be provided by the application:

```
EwaStatus ewaNetHTTPEnd ( EwaNetHandle handle );
```

This function should close the TCP connection corresponding to the request and return `EWA_STATUS_OK` or `EWA_STATUS_OK_YIELD` on success or `EWA_STATUS_ERROR` on failure. Note that returning `EWA_STATUS_OK_YIELD` causes EmWeb/Server to give up the CPU. Otherwise, EmWeb/Server may continue processing other HTTP requests in progress.

If persistent connections are used, it is possible that several HTTP requests will be handled over a single TCP/IP connection. For some applications, it is necessary to reset application-specific state information and to release application-specific dynamic resources after each HTTP request. The following function may be provided by the application for this purpose:

```
void ewaNetHTTPCleanup ( EwaNetHandle handle );
```

This function must be provided by the application or defined as a empty macro. It is invoked by EmWeb/Server when a request completes, allowing the application to reset any processing state stored in the network handle. Note that for persistent connections, this routine may be called several times for the same connection, as one connection can be used for multiple requests. This routine will be called before `ewaNetHTTPEnd()` is invoked.

The EmWeb/Server must have access to an application-provided null-terminated character string representing the HTTP network location (e.g. "www.agranat.com:80". Note that the host name (left of the ':') may be a dotted decimal IP address. Also note that the ':<port-number>' may be omitted if the standard HTTP TCP port #80 is used. This string is returned by the following application-provided function (or macro).

```
const char * ewaNetLocalHostName ( EwsContext context );
```

Copyright © 1997 Agranat Systems, Inc.

4.1.3. Document and Archive Management

The EmWeb/Compiler generates an archive of one or more documents. Documents can be HTML files, JAVA programs, graphical images, or any other information resource addressable by a URL. Archives may be independently loaded or unloaded into the EmWeb/Server.

In most applications, the entire set of available documents are compiled into a single archive and loaded at boot time. However, some applications may desire to dynamically load archives at run-time as needed in order to reduce memory requirements. In fact, applications may implement a scheme similar to virtual memory page swapping under some operating systems to cache an active set of documents in memory while storing other documents in a secondary storage area. Such a secondary storage area could be in FLASH memory, disk, or on a remote server using TFTP or other protocols to load documents at run-time.

An EmWeb archive consists of two components. First, there is the archive data component containing the database of compressed documents, information about how to construct dynamic documents at run-time, access controls, etc. Second, there is the archive object component containing the run-time object code used for the construction of dynamic documents, etc.

Operating systems supporting the run-time loading and linking of object code may off-load both the data and object archive components to a secondary storage area. Otherwise, only the data components are off-loaded while the object components are statically linked into the run-time executable image.

Each archive contains an archive descriptor in the object component. The archive descriptor is referenced by a public symbol generated by the EmWeb/Compiler.

4.1.3.1. Installing and Removing Archives

In order to activate an archive at run-time, the application must invoke `ewsDocumentInstallArchive` with parameters indicating the address of the data component and the archive descriptor of the object component.

```
EwsStatus ewsDocumentInstallArchive
        ( EwsArchive descriptor, const uint8 *datap );
```

This function may return one of the following status codes:

`EWS_STATUS_OK`	The archive was installed successfully
`EWS_STATUS_BAD_MAGIC`	The data or object portion of the archive is invalid or corrupt. This may happen if the EmWeb/Server is configured for a big-endian archive when the EmWeb/Compiler generated a little-endian archive, or vice versa.
`EWS_STATUS_BAD_VERSION`	The archive generated by the EmWeb/Compiler is not understood by the EmWeb/Server.
`EWS_STATUS_ALREADY_EXISTS`	The archive has already been installed.

Copyright © 1997 Agranat Systems, Inc.

EWS_STATUS_IN_USE	The archive contains a document with a URL that has been previously loaded from a different archive.
EWS_STATUS_NO_RESOURCES	Insufficient dynamic memory is available to install the archive.

The archive may be deactivated by invoking ewsDocumentRemoveArchive.

```
EwsStatus ewsDocumentRemoveArchive ( EwsArchive descriptor );
```

These functions return EWS_STATUS_OK on success. Otherwise, EW_STATUS_IN_USE is returned indicating that the archive contains documents that are either being processed by outstanding HTTP requests or that have been cloned to other URLs.

The fixed data component of an archive contains the name of the archive and the date it was created by the EmWeb/Compiler. These values may be retrieved as follows:

```
const char * ewsDocumentArchiveName ( const uint8 *datap );
const char * ewsDocumentArchiveDate ( const uint8 *datap );
const char * ewsDocumentArchiveDate1036 ( const uint8 *datap );
```

4.1.3.2. Demand Loading

In order to implement on-demand archive loading, the application may register document URLs with the EmWeb/Server which are valid but not loaded. This is done by invoking the following function:

```
EwsDocument
ewsDocumentRegister (const char *url, EwaDocumentHandle handle);
```

This function returns an EmWeb/Server document descriptor of type EwsDocument, or EWS_DOCUMENT_NULL on error. The handle is an application-defined value passed back unchanged to the application in ewaDocumentFault as shown below.

If a registered document is requested by a Web browser, then EmWeb/Server notifies the application by invoking the following application-provided function:

```
EwaStatus
ewaDocumentFault ( EwsContext context, EwaDocumentHandle handle );
```

At this point, the application may load a new archive (possibly removing a previously installed archive to make room). When the archive containing the page is installed, the EmWeb/Server automatically completes processing the request when the archive is loaded. The request can be aborted either immediately by returning EWA_STATUS_ERROR from the ewaDocumentFault function, or by invoking ewsNetHTTPAbort.

Once a document is registered, there is no need to re-register it in the event that the

Copyright © 1997 Agranat Systems, Inc.

corresponding archive is removed. EmWeb/Server remembers that the document has been registered as dynamically loadable. However, the application may de-register a document by invoking the following function:

```
EwsStatus ewsDocumentDeregister ( EwsDocument document );
```

This function returns EWS_STATUS_OK on success. Otherwise, EWS_STATUS_IN_USE is returned indicating that the document was cloned or not registered.

4.1.3.3. Cloning

Documents may be cloned and assigned to a new URL. This allows multiple instances of a document to exist while minimizing the storage requirements. An application-specific handle can be used to identify an instance of a document from the request context.

The application clones a document by invoking the following function:

```
EwsDocumentewsDocumentClone
 ( const char *baseurl, const char *newurl, EwaDocumentHandle handle );
```

This function returns a document descriptor for the clone, or EWS_DOCUMENT_NULL on error.

The cloned document may be removed by invoking the following function:

```
EwsStatus ewsDocumentRemove ( EwsDocument document );
```

This function returns one of the following status codes:

EWS_STATUS_OK	The document was successfully removed
EWS_STATUS_NOT_CLONED	The document is not a clone.
EWS_STATUS_IN_USE	The document is itself cloned. The clone of the clone must be removed first.

All clones created from documents in an archive must be removed before the archive can be removed.

Note that the application's document handle is made available to the application in the HTTP request context by invoking the following function:

```
EwaDocumentHandle ewsContextDocumentHandle ( EwsContext context );
```

If the document corresponding to the context was not cloned or registered by the application, then this function returns EWA_DOCUMENT_HANDLE_NULL.

4.1.3.4. URL Rewriting

In some applications, it may be desirable to translate between URLs advertised to the outside world and URLs configured in the archive. The optional URL hook feature

Copyright © 1997 Agranat Systems, Inc.

provides the application with an opportunity to rewrite a requested URL before EmWeb/Server looks up the URL in its archive database. The following function (or macro) may be provided by the application for this purpose:

```
char * ewaURLHook ( EwsContext context, char * url );
```

This function passes the requested URL and request context to the application. The application may simply return url if no rewriting is desired, or it may return an absolute local URL to substitute. The returned URL is looked up in the archive and served as if requested directly. If the application returns NULL, the request is aborted.

This functionality is intended to allow the application to select a subdirectory of documents based upon requested language, content encodings, and other information available from the request context.

Note that this function may rewrite and return any portion of the url string parameter (provided that the string's length is not increased), or return a completely different string buffer instead. If a different buffer is used, the server copies the value immediately.

4.1.3.5. Document Data Access

In some applications, it may be desirable to access the content of a file pointed to by a URL directly (e.g. data files used by CGI scripts, etc.). The following function is optionally provided by the EmWeb/Server to the application for this purpose:

```
EwsStatus ewsDocumentData
        ( const char * url, uint32 *bytesp, const uint8 **datapp );
```

This function looks up the requested URL and, if found, updates `*bytesp` and `*datapp` with the length and pointer to the start of raw data in the file. (Note that the data is compressed if compression is enabled). Since EmWeb employs a proprietary compression technique, the application should specifically disable compression of files it wishes to access this way by using the `nocompress` attribute in the appropriate _ACCESS file (see 3.4. _ACCESS Files, page 22). This function returns EWS_STATUS_OK on success, or EWS_STATUS_NOT_FOUND on failure.

4.1.4. Authentication and Security

HTTP Authorization techniques are discussed in the HTTP specifications (see 7. References, page 87). Knowledge of these principles is essential for correct use of these authorization interfaces.

One or more HTML documents can be associated with a single "realm". A realm is a case-sensitive ASCII string that defines a "protection space" for the associated document(s).

Each realm may have zero or more authentication entries associated with it. If a realm has no authentication entries, then documents in that realm are not accessible.

Copyright © 1997 Agranat Systems, Inc.

A client that attempts to access a document associated with a non-null realm is required to authenticate itself. To authenticate, the client must send authentication parameters valid for at least one authentication entry for that document's realm. Clients that do not authenticate are denied access to the requested document.

For example, assume a realm exists called "foo". It has three authentication entries associated with it (two using the "basic cookie" scheme defined in the HTTP/1.0 specification, and one using the "digest" scheme defined in the HTTP/1.1 specification):

```
REALM: "foo"
        Authentication Entry 1:
             Type: "basic"
             parameters:   Username="user1"
                           Password="guest"
             EwaAuthHandle: <application's entry1 identifier>

        Authentication Entry 2:
             Type: "digest"
             parameters:   Username="user2"
                           Password="837I8U9"
                           DigestRequired=FALSE
             EwaAuthHandle:  <application's entry2 identifier>

        Authentication Entry 3:
             Type: "basic"
             parameters: Username="sinclair"
                         Password="babylon"
             EwaAuthHandle:  <application's entry3 identifier>
```

When a client attempts to access a document associated with realm "foo", it needs to authenticate against one of the above entries. Which one the client authenticates against is at its discretion.

When a client does authenticate against one of the above entries, the `EwaAuthHandle` for that entry is stored in the current context for the HTTP request (`EwsContext`). The datatype of this `EwaAuthHandle` is implementation-defined.

The following types are defined for the authorization application interface (from `src/include/ews_auth.h`):

```
/*
 * Defines the authorization schemes supported by the EmWeb server.
 * New schemes will be added as they are supported.
 */
typedef enum
{
  ewsAuthSchemeBasic,
  ewsAuthSchemeDigest,
  ewsAuthSchemeManualBasic,
  ewsAuthSchemeManualDigest,
```

Copyright © 1997 Agranat Systems, Inc.

```
    ewsAuthMaxScheme            /* count of supported schemes */
  } EwsAuthScheme;

  /*
   * this structure defines parameters for all the supported
   * authorization types.  New parameters will be added in
   * to this structure in the future
   */
  typedef union
  {
  #ifdef EW_CONFIG_OPTION_AUTH_BASIC
    struct
    {
          char  *userName;
          char  *passWord;
    } basic;
  #endif /* EW_CONFIG_OPTION_AUTH_BASIC */
  #ifdef EW_CONFIG_OPTION_AUTH_DIGEST
    struct
    {
          char    *userName;
          char    *passWord;

  #       ifdef EW_CONFIG_OPTION_AUTH_DIGEST_M
          boolean digestRequired; /* require message data verification */
  #       endif /* EW_CONFIG_OPTION_AUTH_DIGEST_M */
    } digest;
  #endif /* EW_CONFIG_OPTION_AUTH_DIGEST */
    char reserved1;
  } EwsAuthParameters;

  /*
   * This structure represents a single authorization entry.
   */
  typedef struct
  {
    EwsAuthScheme     scheme;
    EwsAuthParameters params;
    EwaAuthHandle     handle;     /* user defined */
  } EwsAuthorization;
```

An authorization entry may be created by invoking the following function:

```
EwsAuthHandle ewsAuthRegister
        ( const char *realm, const EwsAuthorization *authorization );
```

This function returns an EmWeb/Server authorization handle which corresponds to the authorization entry, or EWS_AUTH_HANDLE_NULL on failure. The entry contains the triplet (authorization scheme (e.g. basic cookie), scheme-specific parameters (e.g. user and password), application-specific authorization handle).

The application-specific authorization handle corresponding to the authentication validating a particular HTTP request is available from the context as follows:

Copyright © 1997 Agranat Systems, Inc.

```
EwaAuthHandle ewsContextAuthHandle ( EwsContext context );
```

The application may remove a previously registered authorization handle by invoking the following function:

```
EwsStatus ewsAuthRemove ( EwsAuthHandle authorization );
```

This function returns EWS_STATUS_OK on success. Otherwise, an error code is returned (there are presently no conditions that result in error).

By default, documents assigned to a non-null realm are protected and may only be accessed by authorized users. It may be desirable to enable or disable a realm's access controls. The following function disables access control for a realm making documents assigned to the realm accessible to everyone.

```
EwsStatus ewsAuthRealmDisable ( const char * realm );
```

This function returns EWS_STATUS_OK on success, or EWS_STATUS_BAD_REALM if the realm was not defined by any loaded archives.

The following function enables access control for a realm making documents assigned to the realm protected. Protected documents may only be accessed by authorized users.

```
EwsStatus ewsAuthRealmEnable ( const char * realm );
```

This function returns EWS_STATUS_OK on success, or EWS_STATUS_BAD_REALM if the realm was not defined by any loaded archives.

The following function (or macro) must be provided by the application. The function is invoked by EmWeb/Server to get a realm qualifier string from the application. The qualifier string is concatenated with the base realm name in authentication protocol messages such that browsers may differentiate between realms of the same name appearing in multiple servers.

```
const char *ewaAuthRealmQualifier ( void );
```

The application would typically return the string "@hostname", where hostname is the server's local host name or IP address. The application may return NULL if no qualifier is desired.

4.1.4.1. Basic Authentication

The Basic authentication scheme requires the browser client to transmit a Base64 encoded username and password to the server for authentication. While no less secure than SNMP v1, Basic authentication is vulnerable to network monitoring and replay attacks.

The authentication parameters for Basic authentication are simply the username and

Copyright © 1997 Agranat Systems, Inc.

password represented as null-terminated character strings.

4.1.4.2. Manual Basic Authentication

For some applications, it may be impossible to obtain the Basic authentication username/password pair prior to receiving requests for a document. Instead, the Basic authentication username/password may be stored in a remote secure database. This database would need to be queried for the username/password as each request requiring authentication is processed.

In order to support this, a special type of authentication scheme is defined: ewsAuthSchemeManualBasic. This scheme is available only if the build option `EW_CONFIG_OPTION_AUTH_MBASIC` is defined. Once defined, the application must provide the following interface:

```
boolean ewaAuthCheckBasic(EwsContext context
                         ,const char *realm
                         ,const char *basicCookie
                         ,const char *userName
                         ,const char *password );
```

The `ewsAuthSchemeManualBasic` scheme can be registered for any realm by using the `ewsAuthRegister` function. The `scheme` field of the `EwsAuthorization` parameter must be set to `ewsAuthSchemeManualBasic`, and the application handle can be set in the handle field. The params field is not used by the manual basic scheme. Manual basic can only be registered once per realm. If there are "non-manual" (`ewsAuthSchemeBasic`) Basic authorization entries registered to the same realm as a manual Basic entry, the "non-manual" authorization entries will be given preference. If no "non-manual" authorization entries match the request, then `ewaAuthCheckBasic` will be called.

The EmWeb/Server will invoke the application's `ewaAuthCheckBasic` function when authenticating a request for a realm protected by the manual basic scheme, unless a "non-manual" basic authorization entry matched. The application's `ewaAuthCheckBasic` function must now determine whether or not the authorization is valid. This function is passed the context, requested document's realm, and the base64-encoded basic cookie from the Authorization header (as specified in RFC 2068). If the build option `EW_CONFIG_OPTION_AUTH_MBASIC_DECODE` is defined, then `userName` and `password` will point to the username/password strings decoded from the `basicCookie`, otherwise these pointers are `NULL`. All strings are null terminated, and should not be modified by the application.

The return value from `ewaAuthCheckBasic` determines how the server will respond to the request. If `FALSE` is returned, access will be denied. If `TRUE` is returned, access will be granted .

Optionally, the application can suspend the context from within `ewaAuthCheckBasic` using `ewsSuspend`. EmWeb/Server will ignore the return value in this case. When the

Copyright © 1997 Agranat Systems, Inc.

context is later resumed by `ewsResume`, the `ewaAuthCheckBasic` function will be reinvoked by EmWeb/Server.

The application may abort the context from within `ewaAuthCheckBasic` by calling `ewsNetHTTPAbort`.

4.1.4.3. Digest Authentication

The Digest authentication scheme challenges the browser client using a "nonce" value. The client responds with a valid cryptographic checksum of the username, secret password, the given nonce value, the HTTP request method, and the requested URI. This avoids sending the secret password over the network. Furthermore, by generating unique nonce values, the server can make replay and other forms of attack impractical.

> **Note:** EmWeb's digest authentication support is derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm.

> **Note:** Digest authentication is currently an internet proposed standard and is not yet supported by many browsers.

The authentication parameters for Digest authentication include the username and password represented as null-terminated character strings. In addition, one additional boolean parameter is defined as follows:

digestRequired
: If `TRUE`, the EmWeb/Server will require that the client provide a valid message digest to prevent forgery of the request message. This requires significantly more computation and is not supported by most browsers.

The application is required to define the application-specific structure called `EwaAuthNonce`. The `EwaAuthNonce` structure is application-specific and contains parameters to be used in calculating (via MD5 hash of the structure's contents) a unique nonce value. We recommend including the client's IP address, timestamp, a server-wide secret key, and any other random bits the application desires.

Two additional functions must be provided by the application for the generation and verification of nonce values as follows:

First, the `ewaAuthNonceCreate()` function is called by EmWeb/Server in response to an unauthenticated request for a document to request that the application initialize the `EwaAuthNonce` structure.

```
void
ewaAuthNonceCreate
        (EwsContext context
        ,const char *realm
        ,EwaAuthNonce *noncep );
```

The application should initialize the EwaAuthNonce structure. The request context and realm are provided as inputs. For example, a typical application could read the application-specific network handle from the request context to determine the client's IP address.

Second, the `ewaAuthNonceCheck()` function is called by EmWeb/Server in response to a client's request to authenticate against a nonce value previously initialized by the application in `ewaAuthNonceCreate()`.

```
typedef enum EwaAuthNonceStatus_e
  {
    ewaAuthNonceOK,      /* nonce value valid for request */
    ewaAuthNonceLastOK,  /* nonce value valid, but won't be again */
    ewaAuthNonceStale,   /* nonce value is stale, generate new nonce */
    ewaAuthNonceDenied   /* nonce value is invalid */
  } EwaAuthNonceStatus;

EwaAuthNonceStatus
ewaAuthNonceCheck
          (EwsContext context
          ,const char * realm
          ,EwaAuthNonce *noncep,uintf count );
```

This function is called by EmWeb/Server to verify that the nonce is valid for the current request. The `count` parameter indicates the number of times this nonce has been used previously (i.e. zero on the first call). The application may decide to accept the nonce by returning `ewaAuthNonceOK` or `ewaAuthNonceLastOK`. The `ewaAuthNonceLastOK` is used to signal to the server that the nonce value is valid this time, but will not be valid if used again. The server uses this information to generate a new nonce and pass a next nonce value to the browser to be used on the next request. Otherwise, the application may decide to expire the nonce (because it has been used too many times, because too much time has gone by since it was created, or any other reason that the application decides) by returning `ewaAuthNonceStale`, or refuse to authenticate (because the client IP address doesn't match) by returning `ewaAuthNonceDenied`.

4.1.4.4. Manual Digest Authentication

For some applications, it may be impossible to obtain the digest authentication username/password pairs prior to receiving requests for a document. Instead, the digest authentication username/password may be stored on a remote server. This server would need to be queried for the username/password as each request requiring authentication is processed.

In order to support this, a special type of authentication scheme is defined: `ewsAuthSchemeManualDigest`. This scheme is available only if the build option `EW_CONFIG_OPTION_AUTH_MDIGEST` is defined. Once defined, the application must provide the following interface:

Copyright © 1997 Agranat Systems, Inc.

```
boolean ewaAuthCheckDigest( EwsContext context
                           ,const char *realm
                           ,const EwaAuthNonce *noncep
                           ,const char *userName
                           ,const char **digest );
```

The ewsAuthSchemeManualDigest scheme can be registered for any realm by using the ewsAuthRegister function. The scheme field of the EwsAuthorization parameter must be set to ewsAuthSchemeManualDigest, and the application handle can be set in the handle field. The params.digest.digestRequired flag may be used if the EW_CONFIG_OPTION_AUTH_DIGEST_M build option is defined. All other parameters are not used by the manual digest scheme. Manual digest can be registered once per realm. If there are "non-manual" (ewsAuthSchemeDigest) digest authorization entries registered to the same realm as a manual digest entry, the "non-manual" authorization entries will be given preference. If no "non-manual" authorization entries match the request, then ewaAuthCheckDigest will be called.

The EmWeb/Server will invoke the application's ewaAuthCheckDigest function when authenticating a request for a realm protected by the manual digest scheme, unless a "non-manual" digest authorization entry matched. The application's ewaAuthCheckDigest function must now determine whether or not the authorization is valid. This function is passed the context, the requested document's realm, the nonce associated with this request, and the username as it appears in the request. All strings are null terminated, and should not be modified by the application.

ewaAuthCheckDigest returns a boolean status that determines how the server will respond to the request. If FALSE is returned, access will be denied. If TRUE is returned, the *digest parameter has been set to a null terminated string containing the ASCII string representation of the MD5 checksum of the username, realm, and password strings. This is referred to the H(A1) value in RFC 2069. For example, if the username is "user", password is "password", realm is "foo_realm", then *digest should be set to the string representation of MD5(user:foo_realm:password), which would be:

```
*digest = "eafd9339d0114895926c24ec79af7211"
```

The *digest value is used by the server to validate the response digest and entity-digest that are present in the request's Authorization header. If the validation does not pass, access will be denied. The memory used by the *digest value will remain referenced by the server until the context of the request is freed.

Optionally, the application can suspend the context from within ewaAuthCheckDigest using ewsSuspend. The return value will be ignored. When the context is later resumed with ewsResume, the ewaAuthCheckDigest function will be reinvoked.

The application may abort the context from within ewaAuthCheckDigest by calling ewsNetHTTPAbort.

Copyright © 1997 Agranat Systems, Inc.

4.1.4.5. Application Security Verification

Some applications may provide their own proprietary form of client authentication. These proprietary methods will vary from application to application. For example, an application could use the client's IP address to restrict the client's access to certain realms.

EmWeb provides a generic interface that allows an application to provide any proprietary checks on the request context and requested realm. This interface allows the application to deny service to the client if so desired.

This interface is made available by defining the `EW_CONFIG_OPTION_AUTH_VERIFY` build option. The application must then provide the following interface:

```
boolean ewaAuthVerifySecurity( EwsContext context
                              ,const char *realm );
```

This function is called by EmWeb/Server after the current request has passed the standard authorization procedures described above. For example, if basic authentication is enabled on a realm, `ewaAuthVerifySecurity` will be called only after the basic authentication *successfully* completes. If digest authentication is used, then `ewaAuthVerifySecurity` is called after the response digest is *successfully* verified, but before the entity-digest is checked (if the `EW_CONFIG_OPTION_AUTH_DIGEST_M` build option is turned on).

`ewaAuthVerifySecurity` is passed the current context, and a pointer to a null terminated string containing the realm. The application can allow the request to complete by returning `TRUE` from `ewaAuthVerifySecurity`. If `ewaAuthVerifySecurity` returns `FALSE`, the request is denied access.

`ewaAuthVerifySecurity` is provided only as a secondary security mechanism, and can only be used in conjunction with a defined standard security scheme, such as basic or digest.

Optionally, the application can suspend the context from within `ewaAuthVerifySecurity` using `ewsSuspend`. The return value will be ignored. When the context is later resumed, the `ewaAuthVerifySecurity` function will be reinvoked.

The application may abort the context from within `ewaAuthVerifySecurity` by calling ewsNetHTTPAbort.

4.1.4.6. Document Realm Assignment

The authentication realm of a document is typically determined from the `_ACCESS` configuration files by the EmWeb/Compiler. However, it may be desirable to change the document's realm assignment at run-time. The following function is optionally provided by the EmWeb/Server to the application for this purpose:

```
EwsStatus ewsDocumentSetRealm ( const char *url, const char *realm );
```

Copyright © 1997 Agranat Systems, Inc.

This function returns EWS_STATUS_OK on success, or EWS_STATUS_NOT_FOUND if the specified local URL is not installed, or EWS_STATUS_NO_RESOURCES if insufficient resources were available to create a new realm.

4.1.5. Request Context Access

Each HTTP request received by the EmWeb/Server is assigned a unique context. A context handle of type EwsContext is first returned to the application from the ewsNetHTTPStart function.

Application code responsible for the run-time content of HTML documents (i.e. C code fragments from <EMWEB_STRING> and <EMWEB_INCLUDE>, HTML form processing functions ewaFormServe_* and ewaFormSubmit_*, and raw CGI interface functions ewaCGIStart_* and ewaCGIData_*) have access to the corresponding HTTP request context. In the case of <EMWEB_STRING> and <EMWEB_INCLUDE> C code fragments, the context is available from the local variable symbol ewsContext. In the case of the form and raw CGI processing functions, the context is passed as a parameter.

Several context access functions are defined which can be called by the application to extract information about the current context. For example, the function:

```
EwaNetHandle ewsContextNetHandle ( EwsContext context )
```

Returns the network handle corresponding to the HTTP context that was passed by the application to the EmWeb/Server in the ewsNetHTTPStart function. The following table lists these context access functions:

Context Access Function	Description
`ewsContextNetHandle ewsContextNetHandle ( EwsContext context );`	Returns the network handle that was passed by the application to EmWeb/Server in the ewsNetHTTPStart function.
`EwaDocumentHandle ewsContextDocumentHandle ( EwsContext context );`	Returns the document handle that was passed by the application to EmWeb/Server in either the ewsDocumentRegister or ewsDocumentClone functions, or EWA_DOCUMENT_HANDLE_NULL.
`EwaAuthHandle ewsContextAuthHandle ( EwsContext context );`	Returns the authorization handle that was passed by the application to EmWeb/Server in the ewsAuthRegister function, or EWA_AUTH_HANDLE_NULL.

Copyright © 1997 Agranat Systems, Inc.

Context Access Function	Description
`boolean ewsContextIsResuming ( EwsContext context );`	Returns `TRUE` if this is the first application call-out after a suspended request was resumed by calling `ewsResume`. Otherwise, returns `FALSE`.
`uint32 ewsContextIterations ( EwsContext context );`	Returns the iteration count (starting with zero) to an `<EMWEB_STRING>` or `<EMWEB_INCLUDE>` C code fragment with the `EMWEB_ITERATE` attribute specified.
`uintf ewsContextDate ( EwsContext context, char *datap, uintf length );`	Copies the HTTP `Date:` header into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsContextPragma ( EwsContext context, char *datap, uintf length );`	Copies the HTTP `Pragma:` header into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsContextFrom ( EwsContext context, char *datap, uintf length );`	Copies the HTTP `From:` header into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsContextIfModifiedSince ( EwsContext context, char *datap, uintf length );`	Copies the HTTP `If-modified-since:` header into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsContextReferer ( EwsContext context, char *datap, uintf length );`	Copies the HTTP `Referer:` header into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsContextUserAgent ( EwsContext context, char *datap, uintf length );`	Copies the HTTP `User-agent:` value into the buffer provided by the application and returns the actual number of bytes present.

Copyright © 1997 Agranat Systems, Inc.

Context Access Function	Description
`uintf ewsContextHost ( EwsContext context, char *datap, uintf length );`	Copies the HTTP Host: value into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsCGIServerProtocol( EwsContext context, char *datap, uintf length );`	Copies the HTTP server protocol "HTTP/1.0" or "HTTP/0.9" into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsCGIRequestMethod( EwsContext context, char *datap, uintf length );`	Copies the HTTP request method "GET", "HEAD", or "POST" into the buffer provided by the application and returns the actual number of bytes present.
`const char * ewsCGIPathInfo( EwsContext context );`	Returns pointer to CGI extra path information if present, or NULL.
`const char * ewsCGIScriptName( EwsContext context );`	Returns pointer to URL base path string.
`uintf ewsCGIQueryString( EwsContext context, char *datap, uintf length );`	Copies the URL query string into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsCGIContentType( EwsContext context, char *datap, uintf length );`	Copies the HTTP `Content-type:` value into the buffer provided by the application and returns the actual number of bytes present.
`uintf ewsCGIContentLength( EwsContext context );`	Returns the value of the HTTP `Content-length:` header or zero if not present.
`uintf ewsCGIContentEncoding( EwsContext context, char *datap, uintf length );`	Copies the HTTP `Content-encoding:` value into the buffer provided by the application and returns the actual number of bytes present.

Note that the context access functions that copy values into application-provided strings return the actual number of bytes in the value. If the value is not present, then these functions return zero. These functions do not overwrite the application buffer (i.e. the `length` parameter shall be honored by EmWeb/Server). However, the function

Copyright © 1997 Agranat Systems, Inc.

may return a number greater than the `length` if the application buffer was not large enough to accommodate the value. In fact, the `length` parameter of zero may be used to determine the size of the buffer needed.

4.1.6. Raw CGI

We believe that one of the greatest features of EmWeb/Server is that the system integrator can implement nearly any embedded Web-based application without using CGI.

While Agranat Systems strongly discourages its use, we recognize that some highly sophisticated applications may require a raw CGI interface, and so we provide one.

A URL corresponding to the base name of a CGI script may be specified in the _ACCESS file which is processed by the EmWeb/Compiler. The application must provide two functions for each CGI script with names derived from the value of the `cgi-symbol` directive in the _ACCESS file as follows:

```
EwaCGIHandle ewaCGIStart_symbol ( EwsContext context );
```

This function is called by EmWeb/Server when a CGI script is first requested by a Web server. The application returns a handle that is passed back to the application in the second function:

```
void ewaCGIData_symbol ( EwaCGIHandle handle, EwaNetBuffer buffer );
```

This second function is called by EmWeb/Server to pass one or more buffers to the application containing raw CGI data (i.e. the HTTP message body) as they are received from the network. The application is responsible for knowing where the end of data is from the `Content-length:` header (available from `ewsCGIContentLength`).

> **Restriction:** The application's `ewaCGIData_symbol` function must not attempt to access request header information from the request context because the network buffer(s) containing request headers may have been discarded. Any information needed by the application should be copied and saved during `ewaCGIStart_symbol`.

The application may generate a standardized HTTP response header by invoking the following function:

```
EwsStatus ewsCGISendStatus
  ( EwsContext context, const char * status, const char *string);
```

This function sends an HTTP status line for the appropriate HTTP version. The status argument must contain a null-terminated character string containing a three-digit HTTP status code followed optionally by descriptive status text. If not NULL, the string argument contains a null-terminated character string containing additional headers, and/or data. This function returns EWS_STATUS_OK on success. Otherwise, an error

Copyright © 1997 Agranat Systems, Inc.

code (EWS_STATUS_BAD_STATE or EWS_STATUS_NO_RESOURCES) is returned. Note that if ewsCGISendStatus is to be used, it must be used only once per request, and must be used before calling ewsCGIData.

For example, to send a simple HTML page:

```
ewsCGISendStatus ( context, "200 OK",
        "Content-Type: text/html\r\n"
        "Content-Length: 66\r\n"
        "\r\n"
        "<HEAD><TITLE>Example</TITLE></HEAD>"
        "<BODY>This is an example</BODY>" );
```

The application may send additional headers followed by a message body by invoking the following function:

```
EwsStatus ewsCGIData ( EwsContext context, EwaNetBuffer buffer );
```

This function sends the chain of buffers to the Web browser. Note that the application must delimit the header and body sections with a blank line sequence in accordance with the HTTP specification. To complete the request, the application should invoke this function with an EWA_NET_BUFFER_NULL buffer descriptor. This causes EmWeb/Server to close the connection. This function returns EWS_STATUS_OK on success. Otherwise, an error code (EWS_STATUS_BAD_STATE) is returned.

Instead of sending headers and data, the application may instruct EmWeb/Server to generate a redirect to a specified URL or, if local, serve a local document as a response. To send a redirect response to the browser (for a local or remote document) the application may invoke the following function:

```
EwsStatus ewsCGIRedirect ( EwsContext context, const char * url );
```

To serve a local document as a response, the application may invoke the following function:

```
EwsStatus ewsContextSendReply ( EwsContext context, char * url );
```

Note that these functions may not be used in conjunction with ewsCGIData or ewsCGISendStatus. The url parameter may be a relative or absolute URL.

Several CGI environment values are available from the HTTP request context as outlined in the Context section above. In addition, the following two functions return global server information for conformance with CGI/1.0:

```
const char * ewsCGIServerSoftware ( void )
```

Returns the server software version string, for example "Agranat-EmWeb/R1_0".

```
const char * ewsCGIGatewayInterface ( void )
```

Copyright © 1997 Agranat Systems, Inc.

Returns the gateway interface version string, for example "CGI/1.0".

4.1.7. Logging Hook

The application may optionally provide the following function (or macro) for logging HTTP events:

```
void ewaLogHook(EwsContext context, EwsLogStatus status);
```

This function is invoked by EmWeb/Server at least once for each HTTP request, and possibly a second time for certain dispositions after a successful request. Possible status values include (from src/include/ews_def.h):

```
typedef enum EwsLogStatus_e
 {
   /*
    * 200 Request accepted
    */
   EWS_LOG_STATUS_OK,

   /*
    * Request dispositions (after successful request)
    */
  EWS_LOG_STATUS_NO_CONTENT,                  /* 204 no-op form or ima
   EWS_LOG_STATUS_MOVED_PERMANENTLY,            /* 301 link */
   EWS_LOG_STATUS_MOVED_TEMPORARILY,            /* 302 redirect */
   EWS_LOG_STATUS_SEE_OTHER,                    /* 303 see other */
  EWS_LOG_STATUS_NOT_MODIFIED,                /* 304 not modified sinc

   /*
    * 401 Unauthorized
    */
  EWS_LOG_STATUS_AUTH_FAILED,                 /* authorization failed
  EWS_LOG_STATUS_AUTH_FORGERY,                /* bad message checksum
  EWS_LOG_STATUS_AUTH_STALE,                  /* authorization nonce s
  EWS_LOG_STATUS_AUTH_REQUIRED,               /* authorization not pre
  EWS_LOG_STATUS_AUTH_DIGEST_REQUIRED,        /* message digest not pr

   /*
    * 400 Bad Request
    */
   EWS_LOG_STATUS_BAD_REQUEST,                 /* HTTP parse error */
  EWS_LOG_STATUS_BAD_FORM,                    /* form data parse error
  EWS_LOG_STATUS_BAD_IMAGEMAP,                /* imagemap query parse

   /*
    * Additional errors
    */
  EWS_LOG_STATUS_NOT_FOUND,                   /* 404 not found or hidd
  EWS_LOG_STATUS_METHOD_NOT_ALLOWED,          /* 405 method not allowe
  EWS_LOG_STATUS_LENGTH_REQUIRED,             /* 411 length required *
  EWS_LOG_STATUS_UNAVAILABLE,                 /* 503 aborted Document
  EWS_LOG_STATUS_NOT_IMPLEMENTED,             /* 501 bad method for UR
```

```
    EWS_LOG_STATUS_NO_RESOURCES,          /* 500 insufficient reso
    EWS_LOG_STATUS_INTERNAL_ERROR         /* 500 internal error */

} EwsLogStatus;
```

4.1.8. Local Filesystem Interfaces

The EmWeb Server provides the ability to manage a filesystem on the serving system. The Server supports:

- uploading a file from a client (browser) to the server

- serving the contents of a file in response to a URL GET

In both cases, the filesystem is external to the EmWeb Server. EmWeb assumes nothing about the implementation of this filesystem; it can exist on a local disk, non-volatile memory, or on an external proxy. All EmWeb requires is the ability to:

- retrieve/set meta-information about the file (e.g., type, size, modification time, etc.).

- open a file

- read a number of bytes from a existing file

- write a number of bytes to a newly created file

- close a file.

The API that must be supplied to EmWeb to provide these functions is defined below. To enable the basic file support, define the preprocessor symbol `EW_CONFIG_OPTION_FILE` in `ew_config.h`.

4.1.8.1. The EwsFileParams structure

The `EwsFileParams` structure is defined in the `ews_sys.h` include file. It is used by the EmWeb server to associate meta-information with a local file. The `EwsFileParams` structure consists of a union of two substructures. These substructures correspond to the file function that EmWeb is to perform. The `fileField` structure is used when the browser is uploading a file to the server via a form `POST` (see RFC1867). The `fileInfo` structure is used to serve a file in response to a `GET` for a particular URL. The individual fields for each of these structures is described in the description associated file operation.

```
typedef union EwsFileParams_s
{
  /*
   * fileField - for support of the form INPUT TYPE=FILE field.
   * The server fills out this structure, and passes it to
   * the application when the file is submitted
```

```
   */
# ifdef EW_CONFIG_OPTION_FIELDTYPE_FILE
  struct
  {
    const char *fileName;        /* file name or NULL */
    const char *contentType;     /* MIME type */
    const char *contentEncoding; /* content encoding or NULL */
    const char *contentDisposition; /* Content-disposition: */
    int32       contentLength;   /* length or EWS_CONTENT_LENGTH_UNKNOWN */
  } fileField;
# endif /* EW_CONFIG_OPTION_FIELDTYPE_FILE */


  /*
   * fileInfo - for support for local file operations (GET,HEAD,OPTIONS,
   * PUT,DELETE). This structure is setup by the application when a URL
   * that corresponds to a local file is received.  This structure
   * gives the server all it needs to know to handle the file operation.
   */
# if defined( EW_CONFIG_OPTION_FILE_GET )
  struct
  {
    EwaFileName fileName;        /* file name (opaque) */
    const char *contentType;     /* MIME type */
    const char *contentEncoding; /* content encoding or NULL */
    const char *contentLanguage; /* content language or NULL */
    const char *eTag;            /* HTTP/1.1 cachability tag, or NULL */
    const char *lastModified;    /* modification time (RFC1123) or NULL */
    const char *lastModified1036;/* modification time (RFC1036) or NULL */
    const char *realm;           /* auth realm or NULL */
    int32       contentLength;   /* length or EWS_CONTENT_LENGTH_UNKNOWN */
    EwsRequestMethod   allow;    /* allowed methods */
    /* HEAD & OPTION _always_ allowed by default */

  } fileInfo;

  char reserved1;

} EwsFileParams, *EwsFileParamsP;
```

4.1.8.2. File Upload

Support for File Upload is enabled in the EmWeb server by defining the preprocessor symbol `EW_CONFIG_OPTION_FIELDTYPE_FILE` in `ew_config.h`.

To support file upload, EmWeb supports the methods described in RFC1867 (Form-based File Upload in HTML). At the time of this writing, only Netscape Navigator implements this RFC. For more information about RFC1867 and file upload support, contact Agranat Systems.

RFC1867 describes a new form input field of `TYPE-FILE`. This form field results in an input field that allows a user to enter a filename. This filename is the name of a file that is accessible by the browser. On submission of this form, the browser reads the

Copyright © 1997 Agranat Systems, Inc.

file, and sends the file's contents to server as part of the POST request.

This file field takes the following form:

```
<INPUT TYPE=FILE NAME=name VALUE=value>
```

Where *name* is the name of the form field and value is the default filename to display in the form when it is first served. When the EmWeb/Compiler encounters one of these form fields, the following form structure fields are generated:

```
char            *name;
EwaFileHandle   name_handle;
```

The *name* field is used by the form serve function only. It allows the application to override the default file name given when the form is first displayed (note: some browsers do not support a default value - call Agranat Systems for more information). The *name*_handle field is used by the form submit function only. It provides the file handle for the submitted file (see the discussion of ewaFilePost below). The status for name_handle is set to (EW_FORM_RETURNED | EW_FORM_DYNAMIC) on successful file POST, else EW_FORM_FILE_ERROR.

When EmWeb receives the submitted file, it allocates an EwsFileParams structure and initializes the fileField substructure. The fileField substructure is setup using the optional file headers that are sent along with the file. The fileField fields are:

fileName
: Null terminated string containing the name of the file. This is usually supplied as the basename of the file (no path information). This field will be a NULL pointer if the browser does not supply a filename.

contentType
: Null terminated string containing the value of the Content-Type header, or a NULL pointer if the Content-Type header is not present (assumed to be text/plain).

contentEncoding
: Null terminated string containing the value of the Content-Encoding header, or a NULL pointer if the Content-Encoding header is not present.

contentDisposition
: Null terminated string containing the value of the Content-Disposition header, or a NULL pointer if the Content-Disposition header is not present.

contentLength
: set to the number of bytes in the file to be posted. If the Content-Length header is not supplied by the browser, then this field will be set to

Copyright © 1997 Agranat Systems, Inc.

EWS_CONTENT_LENGTH_UNKNOWN.

As soon as EmWeb receives the submitted form containing an INPUT TYPE=FILE field with file content, prior to calling the application's submit function, a call will be made to the application interface ewaFilePost:

```
EwaFileHandle ewaFilePost(
        EwsContext context,
        const EwsFileParams *params );
```

This function is provided by the application. The application defines the appropriate type for EwaFileHandle, and the value for EWA_FILE_HANDLE_NULL, which indicates a NULL EwaFileHandle. The params parameter is initialized by EmWeb as described above, and released by EmWeb on return from ewaFilePost.

ewaFilePost should open a file, and return a EwaFileHandle value that uniquely identifies that file. On error, ewaFilePost can return EWA_FILE_HANDLE_NULL, which will cause EmWeb to discard the contents of the submitted file without discarding the other submitted form fields. The application can also call ewsSuspend to suspend the current context. In this case, ewaFilePost will be reinvoked (with ewsContextIsResuming(context) == TRUE) once the application calls ewsResume on the context. The request can be aborted and the connection closed if ewaFilePost invokes ewsNetHTTPAbort.

After ewaFilePost is called, EmWeb will start calling the file write routine:

```
sintf ewaFileWrite( EwsContext context,
                    EwaFileHandle handle,
                    const uint8 *datap,
                    uintf length );
```

handle is set to the value that was returned by the ewaFilePost call. datap is a pointer to the data to be written, and length is set to the number of bytes that can be written from datap. ewaFileWrite returns the number of bytes written, or < 0 if an error occurred. If < 0 is returned, EmWeb will discard the rest of the incoming file without discarding the other submitted form fields. The application can suspend the current context by calling ewsSuspend from within ewaFileWrite and returning the number of bytes written prior to suspending (can be zero). The application can then resume the context by calling ewsResume. This will cause ewaFileWrite to be reinvoked with ewsContextIsResuming(context) == TRUE and datap and length adjusted by the number returned by the suspended call to ewsFileWrite. The application can also call ewsNetHTTPAbort to abort the entire request, and close the connection.

After the entire file has been received and written, the application's form submit function will be called. If the status field has been set to EW_FORM_RETURNED, then the *name*_handle contains the file handle for the file as returned by ewaFilePost. It is the

responsibility of the form submit function to close the file at this point.

Note well that once a file handle is returned by `ewaFilePost`, it is the responsibility of the application to close that file handle during the form submit function. EmWeb/Server will close the file handle if the request is aborted prior to calling the submit

4.1.8.3. File Serve

The EmWeb Server allows an application to provide a file external to an EmWeb archive for the response to a GET request. This functionality is enabled by defining the preprocessor symbol `EW_CONFIG_OPTION_FILE_GET` in `ew_config.h`.

To serve a file, an application must "mark" a GET request URL as representing a non-archive file. Once this is done, the EmWeb Server will "open" this file, read data from it and serve this data in response to the URL GET. After all data is served, EmWeb Server will close the file.

To "mark" a URL as a local file, the application calls the server interface `ewsContextSetFile` during a URL GET:

```
EwsStatus ewsContextSetFile( EwsContext context,
                             EwsFileParamsP params );
```

`ewsContextSetFile` can be called from:

- `ewaURLHook` once the URL is determined to be a local file entity.
- an application's form submit function prior to calling `ewsContextSendReply`. In this case, the URL passed to `ewsContextSendReply` represents the local file to be served.

Note that both the `ewaURLHook` and an application's form submit function can suspend the current context using `ewsSuspend`. This allows the application to do any asynchronous tasks needed prior to calling `ewsContextSetFile`.

The `fileInfo` substructure of the `param` parameter must be set up by the application prior to calling `ewsContextSetFile`. This structure gives EmWeb Server all the information about the file it needs in order to serve the file as response content. The fields are described below:

EwaFileName fileName
: This field is for use only by the application, EmWeb Server ignores this field. The type `EwaFileName` must be specified by the application.

contentType
: set to a NULL terminated string containing the value for the `Content-Type` header that is to be sent with the file.

Copyright © 1997 Agranat Systems, Inc.

contentEncoding
set to a NULL terminated string containing the value for the Content-Encoding header that is sent with the file. If a NULL pointer, no Content-Encoding header is sent.

contentLanguage
set to a NULL terminated string containing the value for the Content-Language header that is sent with the file. If a NULL pointer, no Content-Language header is sent.

eTag
reserved, must be set to NULL.

lastModified
the last modification date of the file in RFC1123 format as a NULL terminated string. If NULL, it is assumed that the file should be served each time it is requested.

lastModified1036
the last modification date of the file in RFC1036 format as a NULL terminated string. Some browsers cannot correctly recognize the RFC1123 format, so this field should be set in addition to the lastModified field. Call Agranat Systems for more information.

realm
the realm the request should be authenticated against for this file as a NULL terminated string, else NULL if no authorization required. Authorization is only checked when the file is served as a result of calling ewsContextSetFile from ewaURLHook - the realm used to protect the form submission is used implicitly when serving the file as a result of a ewsContextSendReply.

contentLength
Set to the number of bytes in the file, else EWS_CONTENT_LENGTH_UNKNOWN if the file's length is unknown.

allow
reserved, must be set to ewsRequestMethodGet.

EmWeb/Server references the param parameter for the duration of the file operation. As such, this parameter must be accessible throughout the life of the request - beyond the context of the ewaContextSetFile call. It can be safely deallocated by the application during the ewaNetHTTPCleanup call.

Once the URL has been "marked" by the ewsContextSetFile call, EmWeb/Server will open the file using the ewaFileGet interface:

Copyright © 1997 Agranat Systems, Inc.

```
EwaFileHandle ewaFileGet( EwsContext context,
                          const char *url,
                          const EwsFileParamsP params );
```

The `url` parameter is the URL that was "marked" as a file access. `params` is the `EwsFileParams` structure that was passed to `ewsContextSetFile`. The application defines the appropriate type for `EwaFileHandle`, and the value for `EWA_FILE_HANDLE_NULL`, which indicates a `NULL EwaFileHandle`.

`ewaFileGet` should open the file identified by `params` and `url`, and return a `EwaFileHandle` that uniquely represents the file. On error, `ewaFileGet` can return `EWA_FILE_HANDLE_NULL`. This causes the connection to abort. If desired, the application can suspend the current context by calling `ewsSuspend` from within `ewaFileGet`. The context can be resumed at a later time by calling `ewsResume`. `ewaFileGet` will be reinvoked (with `ewsContextIsResuming( context ) == TRUE`).

Once the file has been opened by `ewaFileGet`, EmWeb/Server will read the file by calling `ewaFileRead`:

```
sintf ewaFileRead( EwsContext context,
                   EwaFileHandle handle,
                   uint8 *datap,
                   uintf length );
```

This function should copy up to `length` bytes into the area of memory starting at `datap`. `handle` is set to the file handle returned by `ewaFileGet`. On return, `ewaFileRead` should return the number of bytes written into `datap`, which can be less than `length`. On error, `< 0` should be returned (this causes the request to be aborted and the connection close). This function can suspend by calling `ewsSuspend` and returning the number of bytes written into `datap` (can be zero). When the file described by `handle` has been completely read, this routine must return zero. Returning zero indicates that End-of-File has been reached (except when `ewsSuspend` has been called prior to returning).

Once all file data has been read, EmWeb/Server will close the file by calling ewaFileClose:

```
void ewaFileClose ( EwaFileHandle handle, EwsStatus status );
```

`ewaFileClose` should close the file associated with `handle` (the `EwsFileParams` structure can be safely deallocated at this point). `status` will be set to `EWS_STATUS_OK` on successful completion of the file serve, otherwise it will be set to an error code. EmWeb/Server will call this function to close a file handle during an aborted request prior to calling `ewaNetHTTPEnd`. Note that this function does not support suspension.

Copyright © 1997 Agranat Systems, Inc.

4.2. Application Interface Examples

This section is intended to illustrate the use of application interface functions under various scenarios.

The following tables illustrates the application interfaces used in the simplest EmWeb/Server configuration.

Application->Server (Initialization)	Server -> Application	Description
`ewsInit`	`ewaAlloc` `...`	Application initializes EmWeb/Server. The server may allocate run-time memory resources.

`ewsDocumentInstallArchive`
`...`
`ewsAuthRegister`
`...`
	`ewaAlloc`
	`...`
		Application installs one or more document archives and registers zero or more authenticated users. The server allocates run-time memory resources for its internal databases.

Application->Server (HTTP Transaction)	Server -> Application	Description
`ewsNetHTTPStart`	`ewaAlloc` `...`	Application accepts a new HTTP TCP connection request and informs the EmWeb/Server. The server allocates memory resources for the request.

`ewsNetHTTPReceive`
`...`
	`ewaAlloc`
	`...`
	`ewaNetBufferFree`
	`...`
		Application accepts request data from the network and passes it to the server for processing. The server may allocate memory resources to process the request as needed. Some network buffers received may be released at this point.

Copyright © 1997 Agranat Systems, Inc.

Application->Server (HTTP Transaction)	Server -> Application	Description
ewsRun	<EMWEB_STRING> <EMWEB_INCLUDE> ewaFormServe_* ewaFormSubmit_* ewaCGIStart_* ewaCGIData_* ...	The server invokes the application code attached to the requested document in order to construct a response on-the-fly.
	ewaNetBufferAlloc ... ewaNetHTTPSend (STATUS_OK) ...	The server allocates network buffers as needed, fills them with HTTP response data, and sends them to the application for transmission.
	ewaFree ... ewaNetBufferFree ... ewaNetHTTPEnd	The server releases any remaining resources associated with the HTTP transaction and terminates the request. The application closes the TCP connection at this point.

The next table illustrates a typical HTTP transaction during which the application instructs the EmWeb/Server to yield control of the CPU.

Application->Server (Scheduling)	Server -> Application	Description
ewsNetHTTPStart	ewaAlloc ...	Application accepts a new HTTP TCP connection request and informs the EmWeb/Server. The server allocates memory resources for the request.
ewsNetHTTPReceive ...	ewaAlloc ... ewaNetBufferFree ...	Application accepts request data from the network and passes it to the server for processing. The server may allocate memory resources to process the request as needed. Some network buffers received may be released at this point.
ewsRun	<EMWEB_STRING> <EMWEB_INCLUDE> ewaFormServe_* ewaFormSubmit_* ...	The server invokes the application code attached to the requested document.

Copyright © 1997 Agranat Systems, Inc.

Application->Server (Scheduling)	Server -> Application	Description
ewsSuspend		The application suspends the request. Processing on the request stops until resumed.
ewsResume	<EMWEB_STRING> <EMWEB_INCLUDE> ewaFormServe_* ewaFormSubmit_* ...	The application resumes the request. The server re-invokes the application code attached to the requested document in order to construct a response on-the-fly.
	ewaNetBufferAlloc ewaNetHTTPSend (STATUS_YIELD)	The server allocates and sends a response buffer. The application transmits the data and requests that the server yield the CPU.
ewsRun ...	ewaNetBufferAlloc ewaNetHTTPSend (STATUS_YIELD) ...	The application reschedules the server. The server sends the next buffer.
ewsRun	ewaNetBufferFree ... ewaFree ... ewaNetHTTPEnd	The application reschedules the server. The server releases resources and terminates processing of the request.

The next table illustrates a typical scenario in which the application aborts processing of an incomplete HTTP request.

Application->Server (Abort)	Server -> Application	Description
ewsNetHTTPStart	ewaAlloc ...	Application accepts a new HTTP TCP connection request and informs the EmWeb/Server. The server allocates memory resources for the request.
ewsNetHTTPReceive ...	ewaAlloc ... ewaNetBufferFree ...	Application accepts request data from the network and passes it to the server for processing. The server may allocate memory resources to process the request as needed. Some network buffers received may be released at this point.

Copyright © 1997 Agranat Systems, Inc.

Application->Server (Abort)	Server -> Application	Description
ewsNetHTTPAbort	ewaFree ... ewaNetBufferFree ... ewaNetHTTPEnd	Application requests abort of HTTP transaction in progress. Server releases resources and terminates request.

The next table illustrates the on-demand archive loading feature of EmWeb.

Application->Server (Demand Loading)	Server -> Application	Description
ewsDocumentRegister	ewaAlloc ...	Application registers a URL with EmWeb/Server that is not loaded.
ewsNetHTTPStart	ewaAlloc ...	Application accepts a new HTTP TCP connection request and informs the EmWeb/Server. The server allocates memory resources for the request.
ewsNetHTTPReceive ...	ewaAlloc ... ewaNetBufferFree ...	Application accepts request data from the network and passes it to the server for processing. The server may allocate memory resources to process the request as needed. Some network buffers received may be released at this point.
	ewaDocumentFault	The server recognizes the requested URL to be a registered document and notifies the application.
ewsDocumentInstallArchive		The application loads the archive containing the document.
ewsRun	<EMWEB_STRING> <EMWEB_INCLUDE> ewaFormServe_* ewaFormSubmit_* ewaCGIStart_* ewaCGIData_* ...	The server automatically continues processing the request once the document has been loaded. The server invokes the application code attached to the requested document in order to construct a response on-the-fly.
	ewaNetBufferAlloc ... ewaNetHTTPSend (STATUS_OK) ...	The server allocates network buffers as needed, fills them with HTTP response data, and sends them to the application for transmission.

Copyright © 1997 Agranat Systems, Inc.

Application->Server (Demand Loading)	Server -> Application	Description

	`ewaFree`
	...
	`ewaNetBufferFree`
	...
	`ewaNetHTTPEnd`
		The server releases any remaining resources associated with the HTTP transaction and terminates the request. The application closes the TCP connection at this point.

4.3. Porting Guidelines

The EmWeb/Server is distributed as a directory tree of ANSI C files. In order to assure the best possible upgrade path and support from Agranat Systems, most of these files should not be modified by the system integrator.

To port EmWeb/Server to a specific target environment *foo*, a target-specific `config.foo` configuration directory must be created. `config.foo` contains the target-specific configuration files `ew_types.h`, `ew_config.h`, and `config.mak`. These files can be copied from the examples provided in the `config` directory, and modified as appropriate for the target.

To build the server library `obj.foo/ews.a`, use the following command:

```
make CONFIG=foo server
```

> **Note:** The `configure` script may be used on Unix systems to automatically generate the configuration files for the local Unix development environment target. Typing "make" will build the EmWeb/Compiler, the example archives, and the reference Unix port.

4.3.1. Configuration Header Files

The `src/config/ew_types.h` file contains definitions for base C types used throughout the EmWeb/Server. Most of these are straightforward and generally do not require modification under 32-bit CPU architectures. The one definition which may require modification by the system integrator is the preprocessor symbol `EMWEB_ENDIAN` which is defined to either `EMWEB_ENDIAN_BIG` or `EMWEB_ENDIAN_LITTLE`, and reflects the byte-order of the target processor (Intel processors are generally little endian while Motorola processors are generally big endian). The symbol `EMWEB_ARCHIVE_ENDIAN` indicates the byte-order of the archive generated by the EmWeb/Compiler. By default, the compiler generates big-endian archives unless the `--little` command-line argument is present. Note that it is more efficient if the byte-order of the archive matches the byte-order of the target processor.

The `src/config/ew_config.h` file definitions are expected to be modified extensively for porting EmWeb/Server to a particular hardware platform as follows:

Copyright © 1997 Agranat Systems, Inc.

EW_CONFIG_HTTP_PROTOCOL
Define the level of protocol conformance desired. Must be set to either HTTP_1_0 or HTTP_1_1.

EW_CONFIG_OPTION_DATE
Define to enable generation of HTTP `Date:` headers. This function requires that the application provide the `ewaDate` function. This is required for protocol conformance, but some environments may simply not be able to determine the time-of-day.

EW_CONFIG_OPTION_EXPIRE
Define to enable generation of HTTP `Expire:` headers. The symbol `EWS_CONFIG_OPTION_DATE` must also be defined. If a dynamic document is requested (i.e. a document containing content defined by the application as it is being served), then an `Expire:` header is generated with the current time. Otherwise, no `Expire:` header is generated. The code size may be reduced by not defining this symbol. This is recommended for proper HTTP caching.

EW_CONFIG_OPTION_LAST_MODIFIED
Define to enable generation of HTTP `Last-modified:` headers. The symbol `EWS_CONFIG_OPTION_DATE` must also be defined. If a static document is requested, then the archive creation date is returned to the Web browser. Otherwise, the current time is returned to the Web browser. The code size may be reduced by not defining this symbol. This is recommended for proper HTTP caching.

EW_CONFIG_OPTION_CONDITIONAL_GET
Define to enable parsing of received `If-Modified-Since:` headers to support conditional get functionality defined in the standard. This is recommended for proper HTTP caching.

EW_CONFIG_OPTION_PRAGMA_NOCACHE
Define to enable generation of "`Pragma: no-cache`" HTTP/1.0 headers for documents containing dynamic content.

EW_CONFIG_OPTION_PERSISTENT
Define to enable persistent connections. This is recommended, especially with HTTP/1.1, for optimum network performance. Persistent connections enable the browser to pipeline multiple HTTP requests over a single TCP/IP connection which was not possible in the original HTTP/1.0 specification.

EW_CONFIG_OPTION_CHUNKED_OUT
HTTP/1.1 only. Define to enable the generation of chunked encoded data during transmission of documents containing dynamic elements. This makes maintaining a persistent connection possible under certain

Copyright © 1997 Agranat Systems, Inc.

circumstances and is strongly recommended.

EW_CONFIG_OPTION_CHUNKED_IN
HTTP/1.1 only. Required for protocol conformance. Define to enable parsing of chunked form data received from the browser.

EW_CONFIG_OPTION_METHOD_OPTIONS
Define to enable HTTP/1.1 options method.

EW_CONFIG_OPTION_METHOD_TRACE
Define to enable HTTP/1.1 trace method.

EW_CONFIG_OPTION_CACHE_CONTROL
Define to enable generation of HTTP/1.1 Cache-Control: headers. Recommended for optimum cache control.

EW_CONFIG_OPTION_STRING
Define if the application intends to use the <EMWEB_STRING> feature. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_STRING_TYPED
Define to enable typed EMWEB_STRINGS (i.e. use of EMWEB_TYPE attribute).

EW_CONFIG_OPTION_INCLUDE
Define if the application intends to use the <EMWEB_INCLUDE> feature. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FORM
Define if the application intends to use the EmWeb form processing interfaces. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_IMAGEMAP
Define if the application intends to use EmWeb imagemaps. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI
Define if the application intends to use the raw CGI application interfaces. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_LINK
Define if the application intends to use the `link` attribute in _ACCESS files. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CLONING
Define if the application intends to use the `ewsDocumentClone` application

interface. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_DEMAND_LOADING
Define if the application intends to use the `ewsDocumentRegister/ewaDocumentFault` application interface. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_DOCUMENT_DATA
Define if the application intends to use the `ewsDocumentData` application interface. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_DOCUMENT_SET_REALM
Define if the application intends to use the `ewsDocumentSetRealm` application interface. Otherwise, code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CLEANUP
Define if the application desires to perform a graceful shutdown of the EmWeb/Server by invoking `ewsShutdown`. In some application environments, graceful shutdown may not be necessary as a system restart may be accomplished by a re-boot of the processor. The code size may be reduced by not defining this symbol

EW_CONFIG_OPTION_SCHED
Define if the application intends to use EmWeb's internal scheduler making use of the `ewsRun` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_SCHED_SUSP_RES
Define if the application intends to use the `ewsSuspend/ewsResume` application interfaces. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_SCHED_FC
Define if the application intends to use the `ewsNetFlowControl/UnFlowControl` application interfaces. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_URL_HOOK
Define if the application intends to use the URL rewriting feature by providing the function `ewaURLHook` interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_AUTH
Define to enable support for authentication through the `ewsAuthRegister` application interface. Otherwise, the code size may be reduced by not defining this symbol.

Copyright © 1997 Agranat Systems, Inc.

EW_CONFIG_OPTION_AUTH_BASIC

Define to enable support for the HTTP/1.0 basic cookie authentication method. The symbol EW_CONFIG_OPTION_AUTH must also be defined. The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_AUTH_MBASIC

Define to enable support for the "manual basic" authentication scheme. The code size may be reduced by not defining this symbol,

EW_CONFIG_OPTION_AUTH_DIGEST

Define to enable support for the HTTP/1.1 digest authentication method. The symbol EW_CONFIG_OPTION_AUTH must also be defined. The code size may be reduced by not defining this symbol.

Note: EmWeb's digest authentication support is derived from the RSA Data Security, Inc. MD5 Message-Digest Algorithm. The following legal notice can be found in src/lib/ew_md5c.c:

```
Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All
rights reserved.

License to copy and use this software is granted provided that it
is identified as the "RSA Data Security, Inc. MD5 Message-Digest
Algorithm" in all material mentioning or referencing this software
or this function.

License is also granted to make and use derivative works provided
that such works are identified as "derived from the RSA Data
Security, Inc. MD5 Message-Digest Algorithm" in all material
mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either
the merchantability of this software or the suitability of this
software for any particular purpose. It is provided "as is"
without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this
documentation and/or software.
```

EW_CONFIG_OPTION_AUTH_MDIGEST

Define to enable support for the "manual digest" authentication scheme. The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_AUTH_DIGEST_M

Define to enable support for the digestRequired parameter used in digest authentication. This protects against forged messages using proper authentication credentials. The symbol EW_CONFIG_OPTION_AUTH_DIGEST must also be defined. The code size may be reduced by not defining this symbol.

Copyright © 1997 Agranat Systems, Inc.

EW_CONFIG_OPTION_AUTH_VERIFY
Define to enable support for secondary application-defined authentication. The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_RELEASE_UNUSED
Define to include extra code that attempts to release received network buffers as soon as possible if they do not contain information needed for the duration of an HTTP request. Otherwise, if HTTP headers are typically received in multiple network buffers, the EmWeb/Server may hold on to these buffers longer than necessary.

EW_CONFIG_OPTION_FILE
Define to include local file system support. The code size may be reduced by not defining this symbol

EW_CONFIG_OPTION_FILE_GET
Define to include access to local filesystem for GET method requests. This option requires `EW_CONFIG_OPTION_FILE`. The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_COMPRESS
Define to include extra code required for decompression of compressed archives.

EW_CONFIG_OPTION_CONTEXT_DATE
Define to enable the `ewsContextDate` application interface. The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CONTEXT_PRAGMA
Define to enable the `ewsContextPragma` application interface. The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CONTEXT_FROM
Define to enable the `ewsContextFrom` application interface. The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CONTEXT_IF_MODIFIED_SINCE
Define to enable the `ewsContextIfModifiedSince` application interface. Furthermore, EmWeb/Server honors the HTTP `If-modified-since:` header for static documents by comparing to the archive creation date. (Note that dynamic documents are always served). The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CONTEXT_REFERER
Define to enable the `ewsContextReferer` application interface. Otherwise, the code size may be reduced by not defining this symbol.

Copyright © 1997 Agranat Systems, Inc.

EW_CONFIG_OPTION_CONTEXT_USER_AGENT
Define to enable the `ewsContextUserAgent` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CONTEXT_HOST
Define to enable the ewsContextHost application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_SEND_REPLY
Define to enable the ewsContextSendReply application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_SERVER_SOFTWARE
Define to enable the `ewsCGIServerSoftware` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_GATEWAY_INTERFACE
Define to enable the `ewsCGIGatewayInterface` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_SERVER_PROTOCOL
Define to enable the `ewsCGIServerProtocol` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_REQUEST_METHOD
Define to enable the `ewsCGIRequestMethod` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_PATH_INFO
Define to enable the `ewsCGIPathInfo` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_SCRIPT_NAME
Define to enable the `ewsCGIScriptName` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_QUERY_STRING
Define to enable the `ewsCGIQueryString` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_CONTENT_TYPE
Define to enable the `ewsCGIContentType` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_CONTENT_LENGTH
Define to enable the `ewsCGIContentLength` application interface. Otherwise,

Copyright © 1997 Agranat Systems, Inc.

the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_CGI_CONTENT_ENCODING
Define to enable the `ewsCGIContentEncoding` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_RADIO
Define to enable support for EmWeb HTML form radio buttons. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE
Define to enable support for EmWeb HTML form single option selection boxes. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_SELECT_MULTIPLE
Define to enable support for EmWeb HTML form multiple option selection boxes. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX
Define to enable support for EmWeb HTML form checkbox fields. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_TEXT
Define to enable support for EmWeb HTML form text fields. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_IMAGE
Define to enable support for EmWeb HTML form image inputs. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_UINT
Define to enable support for EmWeb HTML form text fields for unsigned integers. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_INT
Define to enable support for EmWeb HTML form text fields for signed integers. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_HEX_INT
Define to enable support for EmWeb HTML form text fields for hexadecimal integers. Otherwise, the code size may be reduced by not defining this symbol.

Copyright © 1997 Agranat Systems, Inc.

EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING
Define to enable support for EmWeb HTML form text fields for hexadecimal octet strings. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_DOTTEDIP
Define to enable support for EmWeb HTML form text fields for dotted IP addresses. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_DECNET_IV
Define to enable support for EmWeb HTML form text fields for DECnet IV addresses. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_IEEE_MAC
Define to enable support for EmWeb HTML form text fields for IEEE MAC addresses. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC
Define to enable support for EmWeb HTML form text fields for FDDI MAC addresses. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_STD_MAC
Define to enable support for EmWeb HTML form text fields for big or little endian standard MAC addresses. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_OID
Define to enable support for EmWeb HTML form text fields for SNMP object identifiers. Otherwise, the code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_FIELDTYPE_FILE
Define to enable support for RFC 1867 file upload from browser. This option requires EW_CONFIG_OPTION_FILE. The code size may be reduced by not defining this symbol.

EW_CONFIG_OPTION_BROKEN_IMS_EXTRA_DATA
Define to handle non-conformant browsers that place extra data at the end of If-Modified-Since: headers. Otherwise, the code size may be reduced by not defining this symbol.

Copyright © 1997 Agranat Systems, Inc.

EW_CONFIG_OPTION_BROKEN_NEED_OPAQUE
Define to handle non-conformant browsers that require optional opaque field in digest authentication headers. Otherwise, the code size may be reduced by not defining this symbol.

EwaNetBuffer
Define the C type used to represent a buffer descriptor.

EWA_NET_BUFFER_NULL
Define the value of a NULL buffer descriptor used to terminate a chain of buffers or indicate no buffers available.

EwaNetHandle
Define the C type used for the application-defined network handle passed to EmWeb/Server in `ewsNetHTTPStart` and available from the request context by `ewsContextNetHandle`.

EwaDocumentHandle
Define the C type used for the application-defined document handle passed to EmWeb/Server in `ewsDocumentClone` or `ewsDocumentRegister` and available from the request context by `ewsContextDocumentHandle`.

EWA_DOCUMENT_HANDLE_NULL
Define the value for a NULL document handle stored in the context for documents that have not been cloned or registered.

EwaAuthHandle
Define the C type used for the application-defined authorization handle passed to EmWeb/Server in `ewsAuthRegister` and available from the request context by `ewsContextAuthHandle`.

EWS_AUTH_HANDLE_NULL
Define the value for a NULL authorization handle stored in the context for requests that have not been authenticated.

EwaAuthNonce
Define the C structure containing parameters used in generating nonce challenges.

EwaCGIHandle
Define the C type used for the application-defined CGI handle passed to EmWeb/Server from `ewaCGIStart_*` and returned to the application in subsequent `ewaCGIData_*`.

EWA_CGI_HANDLE_NULL
Define the value for a NULL CGI handle stored in the context for CGI

Copyright © 1997 Agranat Systems, Inc.

requests.

EWA_TASK_LOCK()
This application-defined macro is invoked by EmWeb/Server to enter a critical region. If the application is preemptive and multiple threads may access EmWeb application interface functions simultaneously, then this macro should be defined in such a way as to disable preemption.

EWA_TASK_UNLOCK()
This application-defined macro is invoked by EmWeb/Server to leave a critical region entered by `EWA_TASK_LOCK()`.

EMWEB_ERROR()
This application-defined macro is invoked by EmWeb/Server to report serious error conditions, usually a result of improper system integration.

EMWEB_WARN()
This application-defined macro is invoked by EmWeb/Server to report an error condition for which recovery is possible.

EMWEB_TRACE()
This application-defined macro is invoked by EmWeb/Server to trace execution for debugging.

EWA_LOG_HOOK
Define to enable the `ewaLogHook()` application interface. Otherwise, the code size may be reduced by not defining this symbol.

EMWEB_SANITY
Define to enable extra code for checking the consistent use of the API and internal data structures. This is strongly recommended during initial porting and debug. However, the code size may be reduced by not defining this symbol.

EMWEB_HAVE_MEMCPY
Define to use application-provided run-time `memcpy` library function. Otherwise, extra server code is included to implement this functionality.

EMWEB_HAVE_MEMSET
Define to use application-provided run-time `memset` library function. Otherwise, extra server code is included to implement this functionality.

EMWEB_HAVE_SPRINTF
Define to use application-provided run-time `sprintf` library function. Otherwise, extra server code is included to handle conversions from integers to strings (%d, %x).

Copyright © 1997 Agranat Systems, Inc.

EMWEB_HAVE_STRCPY
Define to use application-provided run-time `strcpy` library function. Otherwise, extra server code is included to implement this functionality.

EMWEB_HAVE_STRLEN
Define to use application-provided run-time `strlen` library function. Otherwise, extra server code is included to implement this functionality.

EMWEB_HAVE_STRCMP
Define to use application-provided run-time `strcmp` library function. Otherwise, extra server code is included to implement this functionality.

EwaFileHandle
Define the C type used for the application-defined file handle used in the local filesystem API.

EWA_FILE_HANDLE_NULL
Define the value for a `NULL` file handle as used by the local filesystem API.

EwaFileName
Define the C structure used to represent an application-defined filename parameter as used by the local filesystem API.

The following macros may be defined to override EmWeb/Server defaults as follows:

EWS_FILE_HASH_SIZE
Default: 256. Information about each unique URL loaded from an archive, registered, or cloned, is maintained in an open hash table for fast lookup. This macro may be defined to set the number of entries in the hash table, where each entry contains a pointer value. Smaller values will typically decrease memory usage while increasing search times.

EWS_REALM_HASH_SIZE
Default: 4. Information about each unique realm defined in loaded archives or by the `ewsDocumentSetRealm()` function is maintained in an open hash table for fast lookup. This macro may be defined to set the number of entries in the hash table, where each entry contains a pointer value. Smaller values will typically decrease memory usage while increasing search times.

EWS_NONCE_QUEUE_SIZE
Default: 4. If digest authentication is used, outstanding nonce values are maintained in a circular queue. This macro may be defined to set the maximum number of outstanding nonce values, where each value will contain approximately 48 bytes of state information plus the size of the application-defined `EwaAuthNonce` structure. Larger values are recommended if one-time nonces are to be used and/or many simultaneous

Copyright © 1997 Agranat Systems, Inc.

authenticated requests from different clients are likely.

EWS_HTTP_STATUS_204, EWS_HTTP_STRING_204
Default: "No Content", "\r\n". These macros may be used to override the status line and message body issued for a no content HTTP response. (This status is generated in response to a form submission application function returning NULL, or to an undefined region of an imagemap if no default URL was specified).

EWS_HTTP_STATUS_304, EWS_HTTP_STRING_304
Default: "Not Modified", "\r\n". These macros may be used to override the status line and message body issued for a not modified HTTP response. (This status is generated in response to a conditional GET request if the requested document has not changed since the "If-Modified-Since:" value).

EWS_HTTP_STATUS_400, EWS_HTTP_STRING_400
Default: "Bad Request", "\r\n400 Bad Request\r\n". These macros may be used to override the status line and message body issued for a bad request HTTP response. (This status is generated in response to an HTTP request that could not be parsed).

EWS_HTTP_STATUS_401, EWS_HTTP_STRING_401
Default: "Unauthorized", "\r\n401 Unauthorized\r\n". These macros may be used to override the status line and message body issued for an unauthorized HTTP response. (This status is generated in response to an HTTP request that did not contain proper credentials to access the server).

EWS_HTTP_STATUS_404, EWS_HTTP_STRING_404
Default: "Not Found", "\r\n404 Not Found\r\n". These macros may be used to override the status line and message body issued for a not found HTTP response. (This status is generated in response to an HTTP requests for an unknown or hidden URL).

EWS_HTTP_STATUS_411, EWS_HTTP_STRING_411
Default: "Length Required", "\r\n411 Length Required\r\n". These macros may be used to override the status line and message body issued for an internal error HTTP response. (This status is generated in response to an HTTP/1.1 request containing chunk encoded form data with EW_CONFIG_OPTION_CHUNKED_IN disabled).

EWS_HTTP_STATUS_500, EWS_HTTP_STRING_500
Default: "Internal Error", "\r\n500 Internal Error\r\n". These macros may be used to override the status line and message body issued for an internal error HTTP response. (This status is generated in response to an HTTP request resulting in a detectable internal error. This should not ever occur.)

Copyright © 1997 Agranat Systems, Inc.

EWS_HTTP_STATUS_501, EWS_HTTP_STRING_501
Default: "Not Implemented", "\r\n501 Not Implemented\r\n". These macros may be used to override the status line and message body issued for a not implemented HTTP response. (This status is generated in response to an HTTP request containing a method not supported by the URL).

EWS_HTTP_STATUS_503, EWS_HTTP_STRING_503
Default: "Service Unavailable", "\r\n503 Service Unavailable\r\n". These macros may be used to override the status line and message body issued for a service unavailable HTTP response. (This status is generated in response to an HTTP request for a registered and unloaded document when the application-specific ewaDocumentFault() function aborts the request rather than loading the archive containing the URL).

4.3.2. Application-Provided Functions

The following functions must be provided by the application:

- System functions

  . ewaAlloc
  . ewaFree

- Network functions

  . ewaNetBufferAlloc
  . ewaNetBufferFree
  . ewaNetBufferNextGet
  . ewaNetBufferNextSet
  . ewaNetBufferLengthGet
  . ewaNetBufferLengthSet
  . ewaNetBufferDataGet
  . ewaNetBufferDataSet
  . ewaNetHTTPSend
  . ewaNetHTTPEnd
  . ewaNetHTTPCleanup
  . ewaNetLocalHostName

The following functions may be provided by the application if certain optional EmWeb/Server features are desired.

- System functions

  . ewaDate
  . ewaLogHook

Copyright © 1997 Agranat Systems, Inc.

- Document functions

  . ewaDocumentFault
  . ewaURLHook

- Authentication functions

  . ewaAuthRealmQualifier

- Digest Authentication functions

  . ewaAuthNonceCreate
  . ewaAuthNonceCheck

- Manual Authentication functions

  . ewaAuthCheckBasic
  . ewaAuthCheckDigest
  . ewaAuthVerifySecurity

4.4. Memory Requirements

The following table illustrates the approximate incremental executable image size of the EmWeb/Server for various configuration options enabled compiled using GNU gcc-2.7.0 with -O2 for the Intel 486, the Intel i960, and the Motorola MC68000 architectures. These values are subject to change without notice.

Incremental Option	i486	i960	MC68000
Base EmWeb/Server functionality	7,380	7,496	6,238
<EMWEB_STRING>	180	216	184
Typed <EMWEB_STRING>s	464	464	454
<EMWEB_INCLUDE> *(requires some additional code from <EMWEB_STRING>).*	384	400	312
Base EmWeb HTML form support	3,124	3,168	2,548
Imagemap support	288	320	272
Base raw CGI support	640	656	578
URL Link support	16	32	28
Document cloning	400	400	314
Document on-demand loading	492	520	440

Copyright © 1997 Agranat Systems, Inc.

Incremental Option	i486	i960	MC68000
Document data access	64	72	74
Graceful server shutdown	272	264	220
Request scheduling (yield CPU)	656	624	444
Proxy (ewsSuspend/ewsResume) *(requires request scheduling)*	320	296	242
Network flow control *(requires request scheduling)*	368	288	264
URL rewriting (ewaURLHook)	304	208	164
Base filesystem support	176	224	170
GET/HEAD from local filesystem *(Requires base filesystem sup.)*	1,876	1,480	1,326
Base authentication support	1,168	1,032	934
Basic authentication	952	1,056	824
Digest authentication	5,547	6,732	5,366
Client message digest verification *(requires digest authentication)*	1,335	1,016	742
Manual basic authentication *(requires basic authentication)*	608	472	482
Manual basic decode *(requires manual basic)*	336	352	198
Manual digest authentication *(requires digest authentication)*	384	368	366
Manual authentication verification	144	144	128
Run-time document realm override	0	0	0
Base HTTP/1.1 support	1,284	1,160	972
Persistent connections	883	616	614
Generate chunked encoding *(requires HTTP/1.1)*	828	784	632
Parse chunked encoding *(requires HTTP/1.1)*	855	552	468
OPTIONS method	250	240	192

Copyright © 1997 Agranat Systems, Inc.

Incremental Option	i486	i960	MC68000
TRACE method	288	288	204
Generate Cache-Control headers	122	128	96
Generate Date headers (ewaDate)	41	64	32
Generate Expire headers (ewaDate)	76	176	60
Generate Last-Modified headers	82	112	76
Conditional get support	239	280	248
Generate Pragma headers	35	48	34
Early release of network buffers	224	240	208
Compression	525	584	462
Parse Date headers	64	56	72
Parse Pragma headers	75	64	82
Parse From headers	73	72	82
Parse If-Modified-Since headers	64	64	72
Parse Referer headers	76	64	86
Parse User-Agent headers	79	104	90
Parse Host headers	169	128	124
ewsContextSendReply	626	632	494
ewsCGIServerSoftware	31	40	30
ewsCGIGatewayInterface	24	32	22
ewsCGIServerProtocol	64	56	68
ewsCGIRequestMethod	64	64	68
ewsCGIPathInfo	16	16	16
ewsCGIScriptName	16	16	16
ewsCGIQueryString	64	64	68
ewsCGIContentType	80	64	72
ewsCGIContentLength	16	16	16
ewsCGIContentEncoding	64	64	72

Copyright © 1997 Agranat Systems, Inc.

Incremental Option	i486	i960	MC68000
Form radio buttons	328	304	254
Form single option select boxes	57	64	62
Form multiple option select boxes	128	144	124
Form checkboxes	96	96	92
Form text fields	307	352	256
Form image input fields	93	104	86
Form text fields - unsigned integer	0	0	0
Form text fields - signed integer *(requires some code from unsigned integer)*	98	128	90
Form text fields - hexadecimal int *(requires some code from unsigned integer)*	164	184	166
Form text fields - hex string *(requires some code from unsigned integer)*	706	704	550
Form text fields - dotted IP *(requires some code from unsigned integer)*	354	416	310
Form text fields - DECNet IV *(requires some code from unsigned integer)*	272	352	280
Form text fields - IEEE MAC	144	160	136
Form text fields - FDDI MAC *(requires some code from IEEE MAC)*	48	48	44
Form text fields - Standard MAC *(requires some code from IEEE MAC)*	160	192	168
Form text fields - SNMP Object ID	400	400	340
Form file fileds - file upload *(requires base filesystem support)*	4,372	4,232	3,394
Broken If-Modified-Since Extra	84	128	124

Copyright © 1997 Agranat Systems, Inc.

Incremental Option	i486	i960	MC68000
Broken digest need opaque	16	16	16
Internal sprintf library	656	736	582
Internal strcpy library	112	336	46
Internal strcmp library	128	184	52
Internal strlen library	-160	384	162
Internal memcpy library	160	80	-8
Internal memset library	16	16	-2
Extra sanity checking code	1,248	1,080	1,008
Error reporting	2,498	2,984	2,358
Warning reporting	2,527	2,888	2,350
Debug tracing	1,686	2,392	1,764
Logging hook	704	1,056	778
FULL CONFIGURATION	52,677	55,348	44,740

Note that some of the configurable options listed in the table above depend on pieces of code included from previous options as indicated. The table shows the incremental requirements as each option is added from top to bottom. Therefore, the executable image size for an arbitrary set of options may be different than the sum of the values shown above.

Additional archive memory requirements include

- Compressed file contents and embedded application object code (controlled by application)
- 120 bytes per archive
- 36 bytes per document in archive (plus null-terminated URL string)
- 12 bytes per <EMWEB_STRING>, <EMWEB_INCLUDE>, CGI script, or form field
- null terminated strings for authorization realms, mime types, creation date, etc.
- compression dictionary (if compression used).

Additional run-time memory requirements include:

Copyright © 1997 Agranat Systems, Inc.

- Network buffers (controlled by application). EmWeb/Server typically holds on to received buffers containing HTTP headers of interest for the duration of a request, and typically allocates only one or two buffers before sending them to the application for transmission.

- 1,028 - 1,096 bytes total global EmWeb state (using defaults for hash table sizes which may be overridden by application)

- 176 - 636 bytes per HTTP request (plus 48-52 bytes for first served document, plus 48 bytes for decompression context if compressed)

- 48-52 bytes per nested <EMWEB_INCLUDE> document (plus 48 bytes for decompression context if compressed)

- 40-64 bytes per URL in archive (including clones and registered documents)

- 20-44 bytes per authorization realm plus null-terminated realm name string

- 20-28 bytes per authorization entry plus Base64 or MD5 encoding of user/password

Copyright © 1997 Agranat Systems, Inc.

5. Conformance

The EmWeb/Server implements the HTTP/1.1 and HTTP/1.0 protocols, and MD5 Digest Authentication.

The EmWeb/Server raw CGI capabilities support the functionality of CGI/1.1, but the interfaces are non-standard as they are optimized for the EmWeb/Server environment. In addition, some CGI/1.1 interfaces depend upon the specific integration of EmWeb/Server into the application environment.

The EmWeb/Compiler supports HTML/2.0 Forms with RFC 1867 file upload extensions.

Copyright © 1997 Agranat Systems, Inc.

6. Release History

Release 1.0 (12/23/96)	General Availability
Release 1.1 (02/16/97)	Added network flow-control interfaces
Release 1.2 (02/26/97)	Ported to VxWorks
Release 1.3 (03/07/97)	Ported to pSOS
Release 2.0 (04/14/97)	Added HTTP/1.1 support Added support for typed EMWEB_STRINGs Added EMWEB_STATIC cache-control override Added support for remote authentication databases
Release 2.1 (06/20/97)	Added native filesystem support Added RFC 1867 (file upload from browser) support Added support for EMWEB_TYPE=OBJECT_ID Added support for SNMP Research <mibobj> tags Added ewaNetHTTPCleanup() interfaces

Copyright © 1997 Agranat Systems, Inc.

7. References

[1] T. Berners-Lee and D. Connolly. "Hypertext Markup Language - 2.0." RFC 1866, MIT/W3C, November 1995.T. Berners-Lee and D. Connolly. "Hypertext Markup Language - 2.0." RFC 1866, MIT/W3C, November 1995.

[2] T. Berners-Lee, R. Fielding, and H. Frystyk, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May, 1996.

[3] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[4] R. Braden, "Requirements for Internet Hosts -- Application and Support", RFC1123, October 1989.

[5] M. Horton, "Standard for Interchange of USENET Messages", RFC850, June 1983.

[6] J. Franks, P. Hallam-Baker, J. Hostetler, P. Leach, A. Luotonen, E. Sink, and L. Stewart. "An Extension to HTTP: Digest Access Authentication", RFC 2069, January 1997.

[7] E. Nebel, L. Masinter. "Form-based File Upload in HTML" RFC 1867, Xerox Corporation, November 1995.

Copyright © 1997 Agranat Systems, Inc.

A. EmWeb Application Interface Header Files

This appendix is intended for reference only and is not part of this specification. The C header files listed here are part of the standard EmWeb source distribution and may be updated from time to time.

A-1. Configuration

These sample configuration header files are provided in the EmWeb source distribution. It is expected that these files will be modified by the system integrator in order to port the EmWeb/Server to the target environment. No other files in the EmWeb distribution should be modified in any way.

A-1.1. src/config/ew_types.h

This files contains definitions for base C types used throughout the EmWeb product. In most environments, this file will not need to be modified except possibly for the definition of `EMWEB_ENDIAN` and `EMWEB_ARCHIVE_ENDIAN`. However, for some compilers, it may be necessary for the system integrator to modify additional definitions here.

```
/*
 * ew_types.h,v 1.20 1997/03/31 19:14:25 lawrence Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
 * http://www.agranat.com/
 *
 * EmWeb common types
 *
 */
#ifndef _EW_TYPES_H
#define _EW_TYPES_H

/*
```

```
 * C-TYPES
 * The following basic C-types are defined (see ew_config.h)
 */
#ifndef INT8
#   define INT8 signed char
#endif

#ifndef UINT8
#   define UINT8 unsigned char
#endif

#ifndef INT16
#   define INT16 short int
#endif

#ifndef INT32
#   define INT32 long int
#endif

#ifndef P_INT                   /* an integer the same size as a pointer */
#   define P_INT int
#endif

typedef INT8            int8;           /* signed eight bit integer */
typedef INT16           int16;          /* signed sixteen bit integer */
typedef INT32           int32;          /* signed thirty-two bit integer */
typedef UINT8           uint8;          /* unsigned eight bit integer */
typedef unsigned INT16  uint16;         /* unsigned sixteen bit integer */
typedef unsigned INT32  uint32;         /* unsigned thirty-two bit integer */

/*
 * Volatiles for memory shared with devices
 */
typedef volatile int8   vint8;          /* volatile signed eight bits */
typedef volatile int16  vint16;         /* volatile signed sixteen bits */
typedef volatile int32  vint32;         /* volatile signed thirty-two bits */
typedef volatile uint8  vuint8;         /* volatile signed eight bits */
typedef volatile uint16 vuint16;        /* volatile unsigned sixteen bits */
typedef volatile uint32 vuint32;        /* volatile unsigned thirty-two bits */

/*
 * FAST TYPES
 * The following types offer the most efficient compilation for optimized speed
 * These are never assumed to be more than 16 bits.
 */
typedef int             sintf;          /* signed integer */
typedef unsigned        uintf;          /* unsigned integer */

/*
 * BOOLEAN
 * The boolean type is used primarily as a function return type.  Note that
 * a default representation of TRUE and FALSE are defined.  However, care
 * should be taken to avoid comparing against these values.  For most compilers,
 * TRUE can be any non-zero integer while FALSE is defined as zero.
```

```
 */
#define TRUE            (0 == 0)
#define FALSE           (!TRUE)

typedef uintf           boolean;        /* true or false (non-zero or zero) */

/*
 * POINTER INTEGER
 * The pointer integer is an unsigned integer large enough to accommodate a
 * machine pointer.  This is used primarily for generic pointer arithmetic.
 */
typedef P_INT           pint;           /* unsigned integer for pointer */

/*
 * NULL POINTER
 */
#ifndef NULL
#define NULL            (0)
#endif /* NULL */

/*
 * ENDIAN (don't change these)
 */
#define EMWEB_ENDIAN_BIG        1234
#define EMWEB_ENDIAN_LITTLE     4321

typedef enum Endian_e
{
  ewBigEndian    = EMWEB_ENDIAN_BIG,
  ewLittleEndian = EMWEB_ENDIAN_LITTLE
} Endian;

#if 0
/*
 * EMWEB_ENDIAN - Define as endian-ness of target architecture
 */
#ifndef EMWEB_ENDIAN
#  ifdef HAVE_CONFIG_H
#    include "config.h"
#    ifdef WORDS_BIGENDIAN
#      define EMWEB_ENDIAN EMWEB_ENDIAN_BIG
#    else
#      define EMWEB_ENDIAN EMWEB_ENDIAN_LITTLE
#    endif
#  else
#    pragma error "Define EMWEB_ENDIAN for target"
#  endif
#else
#  pragma error "Define EMWEB_ENDIAN for target"
#endif /* HAVE_CONFIG_H */
#endif /* !TBD!  */

/*
 * EMWEB_ARCHIVE_ENDIAN
```

```
 * Define endian for target archive as generated by EmWeb/Compiler
 */
#define EMWEB_ARCHIVE_ENDIAN    EMWEB_ENDIAN_BIG

/*
 * COMPILER COMPATIBILITY
 *
 * __BEGIN_DECLS precedes external definitions, __END_DECLS follows external
 * definitions.  These could be defined as 'extern "C" {' and '}' respectively
s * for GNU C++ compilers, for example.
 */
#ifdef __cplusplus
#define __BEGIN_DECLS extern "C" {
#define __END_DECLS }
#else /* __cplusplus */
#define __BEGIN_DECLS
#define __END_DECLS
#endif /* __cplusplus */

/*
 * PROTOTYPES
 */
__BEGIN_DECLS
__END_DECLS


#endif /* _EW_TYPES_H */
```

A-1.2. src/config/ew_config.h

This file must be modified by the system integrator to taylor EmWeb/Server to the target environment.

```
/*
 * ew_config.h,v 1.77.2.3 1997/06/15 19:22:31 ian Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
```

```
 *
 * http://www.agranat.com/
 *
 * EmWeb common configuration
 *
 */
#ifndef _EW_CONFIG_H
#define _EW_CONFIG_H

/*
 * Types for sized integers: INT8 INT16 INT32 P_INT
 *         These are guesses as to the correct types to produce the
 *         required sizes for your system.
 *         The configure script will attempt to set these correctly
 *         for the compiler platform; you should set when you copy this
 *         file for each target.
 */
#ifndef INT8
#   define INT8 signed char
#endif

#ifndef INT16
#   define INT16 short int
#endif

#ifndef INT32
#   define INT32 int
#endif

#ifndef P_INT                    /* an integer the same size as a pointer */
#   define P_INT int
#endif

#include "ew_types.h"

/*
 * INCLUDES
 * Any system includes needed by application environment go here
 */
extern void application_tick();
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <signal.h>

/*
 * HTTP PROTOCOL VERSION
 */
#       define HTTP_1_0 1000     /* DO NOT MODIFY */
#       define HTTP_1_1 1001     /* DO NOT MODIFY */
/*
 * Select desired conformance level
 */
```

```
#define EW_CONFIG_HTTP_PROTOCOL HTTP_1_1

/*
 * PROTOCOL OPTIONS (>= HTTP/1.0)
 *   (R) Required, but some embedded systems don't have time-of-day
 *   (O) Optional, but strongly recommended
 */
#define EW_CONFIG_OPTION_DATE           /* (R) Generate Date: */
#define EW_CONFIG_OPTION_EXPIRE         /* (O) Generate Expire: */
#define EW_CONFIG_OPTION_LAST_MODIFIED  /* (O) Generate Last-modified: */
#define EW_CONFIG_OPTION_CONDITIONAL_GET/* (O) Parse If-Modified-Since: */
#define EW_CONFIG_OPTION_PRAGMA_NOCACHE /* (O) Generate Pragma: no-cache */

/*
 * PROTOCOL OPTIONS (>= HTTP/1.1)
 *   (O) Optional, but strongly recommended
 *   (*) May be used with HTTP/1.0
 */
#define EW_CONFIG_OPTION_PERSISTENT     /* (O*) persistent connections */
#define EW_CONFIG_OPTION_CHUNKED_OUT    /* (O) respond with chunked encoding */
#define EW_CONFIG_OPTION_CHUNKED_IN     /* (O) parse chunked encoding */
#define EW_CONFIG_OPTION_METHOD_OPTIONS /* (O*) support OPTIONS method */
#define EW_CONFIG_OPTION_METHOD_TRACE   /* (O*) support TRACE method */
#undef EW_CONFIG_OPTION_METHOD_PUT     /* (*) support PUT operations */
#undef EW_CONFIG_OPTION_METHOD_DELETE  /* (*) support DELETE operations */
#define EW_CONFIG_OPTION_CACHE_CONTROL  /* (O*) Generate Cache-Control: */

/*
 * CONFIGURATION OPTIONS
 *
 * The EmWeb/Server may be customized by the application developer to balance
 * between functionality v.s. memory and CPU requirements.  Each of the options
 * below may be selected (#define) or unselected (#undef).
 */
#define EW_CONFIG_OPTION_STRING         /* emweb_string support */
#define EW_CONFIG_OPTION_STRING_TYPED   /* typed emweb_string support */
#define EW_CONFIG_OPTION_INCLUDE        /* emweb_include support */
#define EW_CONFIG_OPTION_FORM           /* form support */
#define EW_CONFIG_OPTION_IMAGEMAP       /* support for imagemaps */
#define EW_CONFIG_OPTION_CGI            /* raw CGI support */
#define EW_CONFIG_OPTION_LINK           /* link node support */
#define EW_CONFIG_OPTION_CLONING        /* URL cloning support */
#define EW_CONFIG_OPTION_DEMAND_LOADING /* demand document loading support */
#define EW_CONFIG_OPTION_DOCUMENT_DATA  /* direct archive data access support */
#define EW_CONFIG_OPTION_DOCUMENT_SET_REALM /* run-time realm assignment sup. */
#define EW_CONFIG_OPTION_CLEANUP        /* graceful cleanup support */
#define EW_CONFIG_OPTION_SCHED          /* request scheduling support */
#define EW_CONFIG_OPTION_SCHED_SUSP_RES /* suspend/resume request support */
#define EW_CONFIG_OPTION_SCHED_FC       /* flow control support */
#define EW_CONFIG_OPTION_URL_HOOK       /* request URL rewriting hook */
#define EW_CONFIG_OPTION_AUTH           /* authorization support */
#define EW_CONFIG_OPTION_AUTH_BASIC     /* basic authorization support */
#define EW_CONFIG_OPTION_AUTH_DIGEST    /* digest authorization support */
#define EW_CONFIG_OPTION_AUTH_DIGEST_M  /* ", verify client message digest */
```

Copyright © 1997 Agranat Systems, Inc.

```
#define EW_CONFIG_OPTION_AUTH_MBASIC    /* "manual" basic authentication */
#define EW_CONFIG_OPTION_AUTH_MBASIC_DECODE /* do base64 decode for MBASIC */
#define EW_CONFIG_OPTION_AUTH_MDIGEST   /* "manual" digest authentication */
#define EW_CONFIG_OPTION_AUTH_VERIFY    /* manual verification of client */
#define EW_CONFIG_OPTION_COMPRESS       /* archive decompression support */
#define EW_CONFIG_OPTION_RELEASE_UNUSED /* Early release of unused buffers */
#define EW_CONFIG_OPTION_FILE           /* local filesystem support */
#define EW_CONFIG_OPTION_FILE_GET       /* local filesystem GET method */
#define EW_CONFIG_OPTION_FILE_DELETE    /* local filesystem DELETE method */
#define EW_CONFIG_OPTION_FILE_PUT       /* local filesystem PUT method */

/*
 * Options to include support for retrieving HTTP request headers
 */
#define EW_CONFIG_OPTION_CONTEXT_DATE
#define EW_CONFIG_OPTION_CONTEXT_PRAGMA
#define EW_CONFIG_OPTION_CONTEXT_FROM
#define EW_CONFIG_OPTION_CONTEXT_IF_MODIFIED_SINCE
#define EW_CONFIG_OPTION_CONTEXT_REFERER
#define EW_CONFIG_OPTION_CONTEXT_USER_AGENT
#define EW_CONFIG_OPTION_CONTEXT_HOST

/*
 * Option for serving document to browser in response to form or CGI
 */
#define EW_CONFIG_OPTION_CONTEXT_SEND_REPLY

/*
 * Options to include support for CGI access functions
 */
#define EW_CONFIG_OPTION_CGI_SERVER_SOFTWARE
#define EW_CONFIG_OPTION_CGI_GATEWAY_INTERFACE
#define EW_CONFIG_OPTION_CGI_SERVER_PROTOCOL
#define EW_CONFIG_OPTION_CGI_REQUEST_METHOD
#define EW_CONFIG_OPTION_CGI_PATH_INFO
#define EW_CONFIG_OPTION_CGI_SCRIPT_NAME
#define EW_CONFIG_OPTION_CGI_QUERY_STRING
#define EW_CONFIG_OPTION_CGI_CONTENT_TYPE
#define EW_CONFIG_OPTION_CGI_CONTENT_LENGTH
#define EW_CONFIG_OPTION_CGI_CONTENT_ENCODING

/*
 * Options to handle certain form field types (EW_CONFIG_OPTION_FORM must be
 * defined for these to take effect).
 */
#define EW_CONFIG_OPTION_FIELDTYPE_RADIO
#define EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE
#define EW_CONFIG_OPTION_FIELDTYPE_SELECT_MULTIPLE
#define EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX
#define EW_CONFIG_OPTION_FIELDTYPE_TEXT
#define EW_CONFIG_OPTION_FIELDTYPE_IMAGE
#define EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_UINT
#define EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_INT
#define EW_CONFIG_OPTION_FIELDTYPE_HEX_INT
```

Copyright © 1997 Agranat Systems, Inc.

```
#define EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING
#define EW_CONFIG_OPTION_FIELDTYPE_DOTTEDIP
#define EW_CONFIG_OPTION_FIELDTYPE_DECNET_IV
#define EW_CONFIG_OPTION_FIELDTYPE_IEEE_MAC
#define EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC
#define EW_CONFIG_OPTION_FIELDTYPE_STD_MAC
#define EW_CONFIG_OPTION_FIELDTYPE_OID
#define EW_CONFIG_OPTION_FIELDTYPE_FILE

/*
 * Force use of the buffer early release option if support for
 * file input field - needed so we don't buffer the entire
 * incoming file.
 */

#ifdef  EW_CONFIG_OPTION_FIELDTYPE_FILE
#ifndef EW_CONFIG_OPTION_RELEASE_UNUSED
#define EW_CONFIG_OPTION_RELEASE_UNUSED
#endif /* EW_CONFIG_OPTION_RELEASE_UNUSED */
#endif /* EW_CONFIG_OPTION_FIELDTYPE_FILE */

/*
 * Options to handle non-conformant browsers
 *
 * BROKEN_IMS_EXTRA_DATA - Several recent versions of Netscape incorrectly
 * add extra information at the end of an If-Modified-Since: header in the
 * form "; (size = n)".  This is in violation of the HTTP specification and
 * causes EmWeb/Server to effectively ignore the header.  Since so many
 * browsers exhibit this behaviour, this work-around is provided to ignore
 * extra characters after the date string.
 */
#define EW_CONFIG_OPTION_BROKEN_IMS_EXTRA_DATA

/*
 *
 * BROKEN_NEED_OPAQUE - NCSA_Mosaic/2.7b5 and Spyglass_Mosaic/2.11
 * incorrectly require the server to generate an opaque parameter
 * in WWW-Authenticate: headers using Digest authentication.
 */
#define EW_CONFIG_OPTION_BROKEN_NEED_OPAQUE

/*
 * SANITY CHECKING
 *
 * Define EMWEB_SANITY to include extra sanity checking code during initial
 * integration and debugging.  Code size may be reduced by not defining this.
 */
#define EMWEB_SANITY

/*
 * PORT-SPECIFIC DEFINITIONS
 */
#define ewaAlloc malloc
#define ewaFree free
```

```
/*
 * TRACING AND LOGGING
 *
 * The following macros provide tracing and logging facilities using ANSI
 * printf() style arguments.
 */
#define EMWEB_ERROR(x)   printf x
#define EMWEB_WARN(x)    printf x
#define EMWEB_TRACE(x)   /*printf x*/
#define EWA_LOG_HOOK     /* define to include ewaLogHook() interface */

/*
 * NETWORK BUFFERS
 * The network buffer structure is defined by the application and used
 * by the EmWeb/Server.  A network buffer could be an index into an array,
 * a pointer to a buffer descriptor, etc.  EmWeb/Server imposes only the
 * following requirements on the application's buffer implementation:
 *
 *   o Given a buffer, a pointer to the start of data may be determined.
 *   o Given a buffer, the number of bytes of data may be determined or
 *     specified.  The EmWeb/Server will never attempt to increase the
 *     size of a buffer.
 *   o Given a buffer, the next buffer on a singly linked list may be
 *     determined or specified.
 *
 * The EwaNetBuffer structure is used by EmWeb/Server to represents a buffer
 * descriptor.  The value EWA_NET_BUFFER_NULL indicates a NULL buffer used to
 * terminate a buffer chain or as a return value from EwaNetBufferAlloc()
 * indicating no buffers available.
 */
typedef struct EwaNetBuffer_s
  {
    struct EwaNetBuffer_s       * next;
    uint8                       * realdata;
    uint8                       * data;
    uintf                       length;
  } EwaNetBuffer_t, * EwaNetBuffer;

#define EWA_NET_BUFFER_NULL     ((EwaNetBuffer) NULL)

/* Buffer primitives - these may be macros or subroutines */

#define ewaNetBufferLengthGet(buffer)           (buffer)->length
#define ewaNetBufferLengthSet(buffer, len)      (buffer)->length = len
#define ewaNetBufferDataGet(buffer)             (buffer)->data
#define ewaNetBufferDataSet(buffer, datap)      (buffer)->data = datap
#define ewaNetBufferNextGet(buffer)             (buffer)->next
#define ewaNetBufferNextSet(buffer, nxt)        (buffer)->next = nxt

/*
 * Application-specific Network handle
 */
typedef struct EwaNetHandle_s EwaNetHandle_t, *EwaNetHandle;
```

Copyright © 1997 Agranat Systems, Inc.

```
#define EWA_NET_HANDLE_NULL     ((EwaNetHandle) NULL)

/*
 * Application-specific network handle cleanup function (or macro)
 * Invoked by EmWeb/Server after each processing each HTTP request
 * before invoking ewaNetHTTPEnd().
 */
#ifndef EW_CONFIG_OPTION_FILE_GET
#define ewaNetHTTPCleanup( handle ) /* no cleanup necessary */
#endif

/*
 * Application-specific Document handle
 */
typedef void * EwaDocumentHandle;
#define EWA_DOCUMENT_HANDLE_NULL        ((EwaDocumentHandle) NULL)

#ifdef EW_CONFIG_OPTION_AUTH
/*
 * AUTHORIZATION
 *
 * The application may attach an application-specific handle to an entry
 * in the authentication database. .The C-type for this handle is defined
 * by the application below.
 */
typedef void * EwaAuthHandle;           /* user handle to ID entry */
#define EWA_AUTH_HANDLE_NULL    ((EwaAuthHandle) NULL)

#ifdef EW_CONFIG_OPTION_AUTH_DIGEST
/*
 * Application-specific nonce parameters - these are used for Digest
 * authentication to generate one-time challenges.  Recommended values
 * include the client's IP address (derived from the network handle),
 * a time-stamp, a server-side secret value, and other random bits and
 * pieces.  The EmWeb/Server generates an MD5 hash on these values (hiding
 * their semantics) to create the nonce value sent as challenges.
 */
typedef struct EwaAuthNonce_s
  {
    uint32      client_ip;
    uint32      timestamp;
    uint32      up_counter;
#   define      EWA_AUTH_SECRET_SIZE 8
    char        secret[EWA_AUTH_SECRET_SIZE];
  } EwaAuthNonce;

#endif /* EW_CONFIG_OPTION_AUTH_DIGEST */

#endif /* EW_CONFIG_OPTION_AUTH */

/*
 * RAW CGI INTERFACE
 *
 * The application may attach an application-specific handle to a raw CGI
```

```
 * request.  The C-type for this handle is defined by the application below.
 */
typedef void * EwaCGIHandle;
#define EWA_CGI_HANDLE_NULL     ((EwaCGIHandle) NULL)

/*
 * LOCKING
 *
 * The following macros must be defined to disable and enable task preemption
 * to protect critical regions if more than one pre-emptive task may access
 * EmWeb/Server API functions at the same time.
 */
#define EWA_TASK_LOCK()
#define EWA_TASK_UNLOCK()

/*
 * GLOBAL STATE
 */
typedef struct EwsState_s *EwsStateP;
/*
 * The ews_state is a pointer to global state information used internally
 * by EmWeb/Server.  In some environements, it may be desirable to maintain
 * multiple threads of EmWeb/Server within a single memory.  To support this
 * functionality, ews_state may be overridden by a macro that expands at
 * run-time to different values.  The only requirement is that the macro
 * returns type (EwsStateP), and that the address of the pointer (&ews_state)
 * can be determined.  For example:
 *
 * #define ews_state (*((EwsStateP*)someFunction(someArgument)))
 */

/*
 * LIBRARIES
 *
 * The EmWeb/Server implementation uses the following standard C library
 * functions:
 *
 *      sprintf - for converting decimal integers to strings, needed to
 *                generate Content-Length: HTTP headers.
 *
 *      strcpy  - for copying null-terminated strings between the application
 *                and the EmWeb/Server (usually URL names).
 *
 *      strcmp  - for comparing null-terminated strings (comparing URL names).
 *
 *      strlen  - get size of string
 *
 * Most embedded environments have these functions available in libraries.
 * However, EmWeb/Server can be configured to provide its own internal
 * implementations for this purpose.  (Note that a general sprintf
 * implementation is not provided.  Instead, only a conversion routine from
 * long integers to strings is needed.
 *
 * The following macros should be defined (#define) if the corresponding
```

```
 * function is provided by the application environment.
 */
#undef EMWEB_HAVE_SPRINTF
#undef EMWEB_HAVE_STRCPY
#undef EMWEB_HAVE_STRCMP
#undef EMWEB_HAVE_STRLEN
#undef EMWEB_HAVE_MEMCPY
#undef EMWEB_HAVE_MEMSET
#undef EMWEB_HAVE_STRCHR
/* If you have 'index' but not 'strchr', define EMWEB_HAVE_STRCHR and
 *   #define EMWEB_STRCHR(s,c) index(s,c)
 */

#ifdef EW_CONFIG_OPTION_FILE
/*
 * SERVER FILE SUPPORT
 *
 * For support of a server side file system.
 */

typedef struct EwaFileHandle_s *EwaFileHandle;
#define EWA_FILE_HANDLE_NULL ((EwaFileHandle) NULL)

#if defined( EW_CONFIG_OPTION_FILE_GET ) \
 || defined( EW_CONFIG_OPTION_FILE_PUT ) \
 || defined( EW_CONFIG_OPTION_FILE_DELETE )
typedef void *EwaFileName;       /* unused for now */
#endif /* EW_CONFIG_OPTION_FILE_xxx */
#endif /* EW_CONFIG_OPTION_FILE */

#endif /* _EW_CONFIG_H */
```

A-2. Common Header Files

The header files provided here may be included by application software and define the application interfaces to the EmWeb/Server.

A-2.1. src/include/ews_api.h

This is a master include file for EmWeb/Server which, in turn, includes all other application interface header files. We recommend that applications include this file for ease of integration with future releases.

```
/*
 * ews_api.h,v 1.12 1997/05/21 21:55:43 ian Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
```

```
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
 * http://www.agranat.com/
 *
 * EmWeb/Server Application Interface
 *
 */
#ifndef _EWS_API_H
#define _EWS_API_H

#include "ew_types.h"   /* application-defined generic types */
#include "ew_config.h"  /* application-defined configuration */

#include "ews_def.h"    /* general interface definitions */
#include "ews_sys.h"    /* system interfaces */
#include "ews_net.h"    /* network interfaces */
#include "ews_doc.h"    /* document interfaces */
#include "ews_auth.h"   /* authentication interfaces */
#include "ews_cgi.h"    /* CGI interfaces */
#include "ews_ctxt.h"   /* context access interfaces */

#endif /* _EWS_API_H */
```

A-2.2. src/include/ews_def.h

This file contains general definitions used throughout the EmWeb/Server application interface.

```
/*
 * ews_def.h,v 1.26 1997/05/25 23:30:32 giusti Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
```

```
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
 * http://www.agranat.com/
 *
 * EmWeb/Server public definitions
 *
 */
#ifndef _EWS_DEF_H
#define _EWS_DEF_H

typedef const char *EwsConstCharP;

/*
 * Status codes returned to the application by EmWeb/Server
 */
typedef enum EwsStatus_e
  {
    EWS_STATUS_OK,
    EWS_STATUS_BAD_MAGIC,
    EWS_STATUS_BAD_VERSION,
    EWS_STATUS_ALREADY_EXISTS,
    EWS_STATUS_NO_RESOURCES,
    EWS_STATUS_IN_USE,
    EWS_STATUS_NOT_REGISTERED,
    EWS_STATUS_NOT_CLONED,
    EWS_STATUS_NOT_FOUND,
    EWS_STATUS_AUTH_FAILED,
    EWS_STATUS_BAD_STATE,
    EWS_STATUS_BAD_REALM,
    EWS_STATUS_FATAL_ERROR,
    EWS_STATUS_ABORTED
  } EwsStatus;

/*
 * Status codes returned to EmWeb/Server by the application
 */
typedef enum EwaStatus_e
  {
    EWA_STATUS_OK,

#   ifdef EW_CONFIG_OPTION_SCHED
    EWA_STATUS_OK_YIELD,
#   endif

    EWA_STATUS_ERROR

  } EwaStatus;

/*
```

```
 * EmWeb/Server request context handle
 */
typedef struct EwsContext_s * EwsContext;
#define EWS_CONTEXT_NULL        ((EwsContext) NULL)

#ifdef EWA_LOG_HOOK
/*
 * Status codes used for logging requests
 */
typedef enum EwsLogStatus_e
  {
    /*
     * 200 Request accepted
     */
    EWS_LOG_STATUS_OK,

    /*
     * Request dispositions (after successful request)
     */
    EWS_LOG_STATUS_NO_CONTENT,              /* 204 no-op form or imagemap */
    EWS_LOG_STATUS_MOVED_PERMANENTLY,       /* 301 link */
    EWS_LOG_STATUS_MOVED_TEMPORARILY,       /* 302 redirect */
    EWS_LOG_STATUS_SEE_OTHER,               /* 303 see other */
    EWS_LOG_STATUS_NOT_MODIFIED,            /* 304 not modified since */

    /*
     * 401 Unauthorized
     */
    EWS_LOG_STATUS_AUTH_FAILED,             /* authorization failed */
    EWS_LOG_STATUS_AUTH_FORGERY,            /* bad message checksum */
    EWS_LOG_STATUS_AUTH_STALE,              /* authorization nonce stale */
    EWS_LOG_STATUS_AUTH_REQUIRED,           /* authorization not present */
    EWS_LOG_STATUS_AUTH_DIGEST_REQUIRED,    /* message digest not present */

    /*
     * 400 Bad Request
     */
    EWS_LOG_STATUS_BAD_REQUEST,             /* HTTP parse error */
    EWS_LOG_STATUS_BAD_FORM,                /* form data parse error */
    EWS_LOG_STATUS_BAD_IMAGEMAP,            /* imagemap query parse error */

    /*
     * Additional errors
     */
    EWS_LOG_STATUS_NOT_FOUND,               /* 404 not found or hidden */
    EWS_LOG_STATUS_METHOD_NOT_ALLOWED,      /* 405 method not allowed */
    EWS_LOG_STATUS_LENGTH_REQUIRED,         /* 411 length required */
    EWS_LOG_STATUS_UNAVAILABLE,             /* 503 aborted Document Fault */
    EWS_LOG_STATUS_NOT_IMPLEMENTED,         /* 501 bad method for URL */
    EWS_LOG_STATUS_NO_RESOURCES,            /* 500 insufficient resources */
    EWS_LOG_STATUS_INTERNAL_ERROR           /* 500 internal error */

  } EwsLogStatus;
```

```
/*
 * ewaLogHook
 * Application logging hook
 *
 * context - context of request being logged
 * status  - logging status
 */
#ifndef ewaLogHook
extern void ewaLogHook(EwsContext context, EwsLogStatus status);
#endif /* ewaLogHook */

#else /* EWA_LOG_HOOK */
#define ewaLogHook(context, status)
#endif /* EWA_LOG_HOOK */

/*
 * EmWeb/Server archive descriptor.
 */
typedef struct EwsArchive_s * EwsArchive;

/*
 * EmWeb/Server document handle
 */
typedef struct EwsDocument_s * EwsDocument;
#define EWS_DOCUMENT_NULL       ((EwsDocument) NULL)

/*
 * EmWeb/Server opaque read-only data handle
 */
typedef const uint8 EwsArchiveData[];

/*
 * EmWeb/Server authorization handle
 */
typedef struct EwsAuthHandle_s * EwsAuthHandle;
#define EWS_AUTH_HANDLE_NULL    ((EwsAuthHandle) NULL)

#ifdef EW_CONFIG_OPTION_FORM
/*
 * Application form interface constants for status byte
 */
#define EW_FORM_INITIALIZED     0x01
#define EW_FORM_DYNAMIC         0x02
#define EW_FORM_RETURNED        0x10
#define EW_FORM_PARSE_ERROR     0x20
#define EW_FORM_FILE_ERROR      0x20        /* equivalent to parse error */

/*
 * EmWeb/Server form element types
 */
#ifdef EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING
typedef struct EwsFormFieldHexString_s
  {
    uint32 length;
```

```
    uint8 *datap;
  } EwsFormFieldHexString, *EwsFormFieldHexStringP;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING */

#if (defined(EW_CONFIG_OPTION_FIELDTYPE_IEEE_MAC)       \
     || defined(EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC)    \
     || defined(EW_CONFIG_OPTION_FIELDTYPE_STD_MAC)     \
     )
typedef uint8 EwsFormFieldMAC[6];
#endif /* MAC */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_OID
typedef struct EwsFormFieldObjectID_s
  {
    uint32 length;      /* number of uint32's in datap */
    uint32 *datap;      /* array of uint32's representing OID */
  } EwsFormFieldObjectID, *EwsFormFieldObjectIDP;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_OID */

#if defined(EW_CONFIG_OPTION_FIELDTYPE_RADIO) ||        \
    defined(EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE)
/*
 * ewsFormEnumToString
 * Lookup string corresponding to enum from context
 *
 * context      - context of request
 * enum         - value of enumerator
 *
 * Returns string from corresponding HTML VALUE= field, or NULL if out of
 * bounds.
 */
extern const char * ewsFormEnumToString( EwsContext context, int value );
#endif /* EW_CONFIG_OPTION_FIELDTYPE_RADIO | SELECT_SINGLE */

#endif /* EW_CONFIG_OPTION_FORM */

/*
 * EwsRequestMethod
 *
 * This is an enum for supported request methods.  Bit values are used so
 * groups of valid request methods can be grouped into a single field
 */
typedef enum EwsRequestMethod_e
  {
    ewsRequestMethodUnknown = 0x0000    /* all others */
   ,ewsRequestMethodGet     = 0x0001    /* GET */
   ,ewsRequestMethodPost    = 0x0002    /* POST */
   ,ewsRequestMethodHead    = 0x0004    /* HEAD */

#  ifdef EW_CONFIG_OPTION_METHOD_OPTIONS
   ,ewsRequestMethodOptions = 0x0008    /* OPTIONS */
#  endif
#  ifdef EW_CONFIG_OPTION_METHOD_TRACE
   ,ewsRequestMethodTrace   = 0x0010    /* TRACE */
```

```
#  endif
#  ifdef EW_CONFIG_OPTION_METHOD_PUT
   ,ewsRequestMethodPut      = 0x0020     /* PUT */
#  endif
#  ifdef EW_CONFIG_OPTION_METHOD_DELETE
   ,ewsRequestMethodDelete   = 0x0040     /* DELETE */
#  endif

  } EwsRequestMethod;



#endif /* _EWS_DEF_H */
```

A-2.3. src/include/ews_sys.h

This file contains definitions and prototypes of the system functions of the application interface.

```
/*
 * ews_sys.h,v 1.17.2.3 1997/06/02 23:46:39 giusti Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
 * http://www.agranat.com/
 *
 * EmWeb/Server system interfaces
 *
 */

#ifndef _EWS_SYS_H
#define _EWS_SYS_H

#include "ew_types.h"
#include "ew_config.h"
#include "ews_def.h"
```

```
/*****************************************************************************
 *
 * INITIALIZATION AND SHUTDOWN
 *
 * The EmWeb/Server must be initialized at start-up before accepting requests,
 * installing document archives, or registering authentication information.
 *
 *****************************************************************************/

/*
 * ewsInit
 * Initialize EmWeb/Server
 *
 * Returns EWS_STATUS_OK on success, else failure code (TBD).
 */
extern EwsStatus ewsInit ( void );

#ifdef EW_CONFIG_OPTION_CLEANUP
/*
 * ewsShutdown
 * Graceful shutdown of EmWeb/Server terminating all requests in progress
 * and releasing all memory and buffers.
 *
 * Returns EWS_STATUS_OK on success, else failure code (TBD).
 */
extern EwsStatus ewsShutdown ( void );
#endif /* EW_CONFIG_OPTION_CLEANUP */

#ifdef EW_CONFIG_OPTION_SCHED
/*****************************************************************************
 *
 * SCHEDULING
 *
 * The EmWeb/Server is capable of both multi-threading HTTP requests and
 * yielding execution control to the CPU.  The following functions are provided
 * to give the application control over how EmWeb/Server schedules  pending
 * requests.
 *
 *****************************************************************************/

/*
 * ewsRun
 * Reschedule request processing after control returned to application as a
 * result of returning EWA_STATUS_OK_YIELD from ewaNetHTTPSend(),
 * ewaNetHTTPEnd(), or ewaDocumentFault(), or as the value to the status
 * parameter in ewsResume().
 *
 * Returns EWS_STATUS_OK on success, else failure code (TBD).
 */
extern EwsStatus ewsRun ( void );

#ifdef EW_CONFIG_OPTION_SCHED_SUSP_RES
/*
```

```
 * ewsSuspend
 * Suspend processing of the current request during an EmWeb callout function
 * (e.g. EmWebString, EmWebFormServe, or EmWebFormSubmit).  This function is
 * used in the implementation of proxies.
 *
 * Note that this does _not_ cause the server to stop processing requests; in
 * fact, it forces a flush of any partially filled buffer for the current
 * request and may process other pending requests.  This just causes the
 * current callout to be marked as 'suspended'.  When the current callout
 * returns, the result will be ignored - after ewsResume (below) is called
 * for this same context, the callout will be repeated.
 *
 * context      - context of request to be suspended
 *
 * Returns EWS_STATUS_OK on success, else failure code (TBD).
 */
extern EwsStatus ewsSuspend ( EwsContext context );

/*
 * ewsResume
 * Resume processing of a request previously suspended by ewsSuspend().  This
 * causes the request to be rescheduled, and the callout from which ewsSuspend
 * had been called will be reinvoked.
 *
 * context      - Context of suspended request to be resumed
 * status       - EWA_STATUS_OK or EWA_STATUS_OK_YIELD.
 *
 * Returns EWS_STATUS_OK on success, else failure code (TBD).
 */
EwsStatus ewsResume ( EwsContext context, EwaStatus status );
#endif /* EW_CONFIG_OPTION_SCHED_SUSP_RES */

#endif /* EW_CONFIG_OPTION_SCHED */

/*****************************************************************************
 *
 * MEMORY MANAGEMENT
 *
 * The EmWeb/Server presumes that the target O/S has a dynamic memory
 * management capability roughly equivalent to the POSIX malloc() and free()
 * functions.
 *
 * Alternatively, the application could create a static linked-list of memory
 * blocks of different sizes.  Refer to the product documentation for
 * information about run-time memory requirements.
 *
 * The application is responsible for providing the following memory management
 * functions to the EmWeb/Server.
 *
 *****************************************************************************/

#ifndef ewaAlloc
/*
 * ewaAlloc
```

```
 * Allocate a block of memory of at least the requested number of bytes.
 *
 * bytes        - requested size of the memory block in bytes
 *
 * Returns a pointer to the first byte of memory, or NULL if no memory of the
 * requested size is available.
 */
extern void * ewaAlloc ( uintf bytes );
#endif

#ifndef ewaFree
/*
 * ewaFree
 * Free a block of memory previously allocated by ewaAlloc()
 *
 * pointer      - pointer to beginning of memory block (from ewaAlloc())
 *
 * No return value
 */
extern void ewaFree ( void * pointer );
#endif

#ifdef EW_CONFIG_OPTION_DATE

/*****************************************************************************
 *
 * TIME-OF-DAY MANAGEMENT
 *
 * If DATE support is enabled, the EmWeb/Server invokes the following
 * application-provided function to retrieve the current date for use in
 * Date: HTTP headers.  Furthermore, the current date is also used in
 * Expire: and Last-modified: headers of dynamically generated documents
 * (i.e. HTML documents containing EMWEB extensions), if configured.
 * Note that the Last-modified: header of static documents use the archive
 * creation date stored in the archive by the EmWeb/Compiler.
 *
 *****************************************************************************/

#ifndef ewaDate
/*
 * ewaDate
 *
 * Return a string containing the current time-of-day in one of the HTTP
 * standard representations as follows:
 *
 * RFC1123: (Preferred standard)
 *   Fri, 28 Jun 1996 15:57:28 GMT
 *
 * RFC850: (Used, but not preferred)
 *   Friday, 28-Jun-96 15:57:28 GMT
 *
 * asctime: (Allowed, but discouraged)
 *   Jun  6 15:57:28 1996
 */
```

```
extern const char * ewaDate ( void );
#endif

#endif /* EW_CONFIG_OPTION_DATE */

#ifdef EW_CONFIG_OPTION_URL_HOOK

/*****************************************************************************
 *
 * URL REWRITING
 *
 * If URL rewriting support is enabled, the EmWeb/Server will invoke the
 * following function when each request has been successfully parsed, but
 * before the URL is looked up in the filesystem.  The application can then
 * return a new (or the same) URL to actually serve.
 *
 *****************************************************************************/

#ifndef ewaURLHook
/*
 * ewaURLHook
 *
 * context      - context of request
 * url          - requested URL
 *
 * Returns new URL, or NULL to cause abort.  (For no-op, simply return url).
 *
 * Note: the application MUST NOT invoke ewsNetHTTPAbort from here!  Return
 * a NULL pointer instead!
 *
 * The URL may be rewritten in place if the length is not increased;
 * if the returned pointer is not within the url parameter, the value is
 * copied by the server.
 */
extern char * ewaURLHook ( EwsContext context, char *url );
#endif

#endif /* EW_CONFIG_OPTION_LOG_HOOK */

/*
 * Since content length of zero is valid, we cannot use zero to
 * indicate when the content length is unknown (eg. dynamic data).
 * Assuming nobody will ever try to send/receive a 4gig document is
 * probably safe for now...
 */
#define EWS_CONTENT_LENGTH_UNKNOWN ((int32) -1)

#ifdef EW_CONFIG_OPTION_FILE
/*****************************************************************************
 *
 * LOCAL FILE SYSTEM SUPPORT
 *
 * The following routines must be supplied to support the server's
 * local file system.
```

```
 */

typedef union EwsFileParams_s
{
  /*
   * fileField - for support of the form INPUT TYPE=FILE field.
   * The server fills out this structure, and passes it to
   * the application when the file is submitted
   */
# ifdef EW_CONFIG_OPTION_FIELDTYPE_FILE
  struct
  {
    const char *fileName;        /* file name or NULL */
    const char *contentType;     /* MIME type */
    const char *contentEncoding; /* content encoding or NULL */
    const char *contentDisposition; /* Content-disposition: */
    int32       contentLength;   /* length or EWS_CONTENT_LENGTH_UNKNOWN */
  } fileField;
# endif /* EW_CONFIG_OPTION_FIELDTYPE_FILE */


  /*
   * fileInfo - for support for local file operations (GET,HEAD,OPTIONS,
   * PUT,DELETE).  This structure is setup by the application when a URL
   * that corresponds to a local file is received.  This structure
   * gives the server all it needs to know to handle the file operation.
   */
# if defined( EW_CONFIG_OPTION_FILE_GET ) \
  || defined( EW_CONFIG_OPTION_FILE_PUT ) \
  || defined( EW_CONFIG_OPTION_FILE_DELETE )
  struct
  {
    EwaFileName fileName;        /* file name (opaque) */
    const char *contentType;     /* MIME type */
    const char *contentEncoding; /* content encoding or NULL */
    const char *contentLanguage; /* content language or NULL */
    const char *eTag;            /* reserved, MUST BE NULL */
    const char *lastModified;    /* modification time (RFC1123) or NULL */
    const char *lastModified1036;/* modification time (RFC1036) or NULL */
    const char *realm;           /* auth realm or NULL */
    int32       contentLength;   /* length or EWS_CONTENT_LENGTH_UNKNOWN */
    EwsRequestMethod   allow; /* reserved, MUST BE ewsRequestMethodGet */
    /* HEAD & OPTION _always_ allowed by default */

  } fileInfo;
# endif /* local file system GET/PUT/DELETE */

  char reserved1;                /* prevent NULL union */

} EwsFileParams, *EwsFileParamsP;


/*
 * ewaFileClose
```

```
 * Closes the file and indicates success or failure
 */
extern void
ewaFileClose ( EwaFileHandle handle, EwsStatus status );

/*
 * ewaFileWrite
 * Write data to a file
 */
extern sintf
ewaFileWrite ( EwsContext context
             , EwaFileHandle handle     /* handle from ewaFileOpen */
             , const uint8  *datap      /* pointer to data */
             , uintf         length     /* length of data */
             );
/* returns bytes written, <0 on error */

/*
 * ewaFileRead
 * Read data from a  file
 */
extern sintf
ewaFileRead ( EwsContext context
            , EwaFileHandle handle    /* handle from ewaFileOpen */
            , uint8 *datap           /* pointer to data buffer */
            , uintf length           /* length of buffer */
            );
/* returns bytes read, 0 on EOF, <0 on error */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_FILE
/*
 * ewaFilePost
 * For multipart/form-data file submissions.  Before the application's
 * submit function is called, the server may invoke this function for
 * each received file as part of a <INPUT TYPE=FILE> element in the
 * form.  If successful, the file handle will be passed to the
 * application's submit function at which point it is the application's
 * responsibility to close the file.
 */
extern EwaFileHandle
ewaFilePost ( EwsContext context
            ,const EwsFileParams *params
            );
/* returns EWA_FILE_HANDLE_NULL on error */
#endif /* EW_CONFIG_OPTION_FIELDTYPE_FILE */

#ifdef EW_CONFIG_OPTION_FILE_GET
/*
 * ewaFileGet
 * When the server determines that the file set by ewsContextSetFile
 * should be served to the browser, this function is invoked to open
 * the file for reading.
 *
 * To support Conditional GET, the request message may include
```

```
 * If-Modified-Since, If-Unmodified-Since, If-Match,
 * If-None-Match, or If-Range headers.  In this case, the
 * server will use the information given in the returned
 * params block to determine if the file should be retrieved.
 */
extern EwaFileHandle
ewaFileGet ( EwsContext context
            ,const char *url
            ,const EwsFileParams *params /* from ewsContextSetFile() */
           );
/* returns EWA_FILE_HANDLE_NULL on error */
#endif /* EW_CONFIG_OPTION_FILE_GET */

#endif /* EW_CONFIG_OPTION_FILE */

#endif /* _EWS_SYS_H */
```

A-2.4. src/include/ews_net.h

This file contains the definitions and prototypes used by the networking functions of the application interface.

```
/*
 * ews_net.h,v 1.20.2.1 1997/06/15 03:03:18 giusti Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
 * http://www.agranat.com/
 *
 * EmWeb/Server application interface to network transport layer
 *
 */
#ifndef _EWS_NET_H
#define _EWS_NET_H

#include "ew_types.h"
#include "ew_config.h"
```

```
#include "ews_def.h"

/*****************************************************************************
 *
 * In a typical TCP/IP implementation, the application is responsible for
 * listening to the HTTP TCP port (80) for connection requests.  When a
 * connection request is received, the application accepts the connection
 * on behalf of EmWeb/Server and invoke ewsNetHTTPStart() to inform the
 * EmWeb/Server of the new request.
 *
 * EmWeb/Server assumes that the application maintains data buffers for
 * the reception and transmission of TCP data.  The only requirements that
 * EmWeb/Server imposes on the buffer implementation is as follows:
 *
 *   1. Buffers can be uniquely identified by a buffer descriptor.  No
 *      assumptions are made about the actual structure of the buffer
 *      descriptors or their relationship to data.  For example, a buffer
 *      descriptor could be an index into a table, a pointer to a
 *      structure (either contiguous or seperate from the data represented),
 *      etc.  The application is responsible for defining the appropriate
 *      type for EwaNetBuffer and value for EWA_NET_BUFFER_NULL.
 *
 *   2. Buffers can be chained together.  Given a buffer descriptor,
 *      EmWeb/Server must be able to get or set the "next buffer in the chain"
 *      field.  This is done by ewaNetBufferNextSet() and
 *      ewaNetBufferNextGet().  Note that the buffer chain is terminated when
 *      the next buffer value is EWA_NET_BUFFER_NULL.
 *
 *   3. Given a buffer descriptor, EmWeb/Server can determine the start of
 *      data in the buffer.  Additionally, EmWeb/Server may change the start
 *      of data in the buffer (EmWeb/Server only changes the start of data in
 *      the buffer upward).  This is done by ewaNetBufferDataGet() and
 *      ewaNetBufferDataSet().
 *
 *   4. Given a buffer descriptor, EmWeb/Server can determine the size of
 *      contiguous data in the buffer.  Additionally, EmWeb/Server may
 *      change the size of the buffer (EmWeb/Server only changes the
 *      size of the buffer downward).  This is done by ewaNetBufferLengthGet()
 *      and ewaNetBufferLengthSet().
 *
 *   5. EmWeb/Server may allocate a buffer by invoking ewaNetBufferAlloc().
 *      If no buffers are available, this function returns EWA_NET_BUFFER_NULL.
 *      Additionally, EmWeb/Server may release a buffer by invoking
 *      ewaNetBufferFree().
 *
 * As the application receives TCP data on an HTTP connection, it passes
 * this data to the EmWeb/Server by invoking ewsNetHTTPReceive().
 *
 * The EmWeb/Server transmits TCP data on an HTTP connection by invoking
 * ewaNetHTTPSend().  The application may throttle EmWeb/Server by
 * returning EWA_STATUS_OK_YIELD.  This causes the EmWeb/Server to
 * save state and return control to the application.  The application must
 * invoke ewsRun() to give the EmWeb/Server an opportunity to continue
 * processing the request.
```

```
 *
 * When the EmWeb/Server completes a request, it invokes ewaNetHTTPEnd().
 *
 * The application may abort a request at any time by invoking
 * ewsNetHTTPAbort().
 *
 ************************************************************************/

/*
 * ewsNetHTTPStart
 * Start a new HTTP request
 *
 * net_handle    - application-specific handle representing request
 *
 * Returns context for the request, or EWS_CONTEXT_NULL on failure.
 */
extern EwsContext ewsNetHTTPStart ( EwaNetHandle net_handle );

/*
 * ewsNetHTTPAbort
 * Abort a previously started HTTP request.
 *
 * context       - context of request to be aborted
 *
 * Returns EWS_STATUS_OK on success, else error code (TBD)
 */
extern EwsStatus ewsNetHTTPAbort ( EwsContext context );

/*
 * ewsNetHTTPReceive
 * Receive request data from the network.
 *
 * context       - context of request to which received data applies
 * buffer        - buffer containing received data
 *
 * Returns EWS_STATUS_OK on success, else error code (TBD)
 */
extern EwsStatus ewsNetHTTPReceive ( EwsContext context, EwaNetBuffer buffer );

#ifdef EW_CONFIG_OPTION_SCHED_FC

/*
 * ewsNetFlowControl
 * Mark context for flow control to avoid predicted congestion.  ewsRun() will
 * place the context on the suspended list at the next opportunity and
 * continue by processing additional requests.
 *
 * context       - context of request to be flow controlled
 *
 * Returns EWS_STATUS_OK.
 */
EwsStatus ewsNetFlowControl ( EwsContext context ) ;

/*
```

```
 * ewsNetUnFlowControl
 * Resume previously flow controlled context
 *
 * context      - context of request to be flow controlled
 *
 * Returns EWS_STATUS_OK.
 */
EwsStatus ewsNetUnFlowControl ( EwsContext context );

#endif /* EW_CONFIG_OPTION_SCHED_FC */

#ifndef ewaNetHTTPSend
/*
 * ewaNetHTTPSend (Application)
 * This function must be provided by the application to accept data from
 * EmWeb/Server for transmission to a browser in response to a request.
 *
 * net_handle   - application-specific request handle from ewsNetHTTPStart()
 * buffer       - buffer containing data to be transmitted
 *
 * Returns EWA_STATUS_OK on success, EWA_STATUS_OK_YIELD on success and
 * request EmWeb/Server to yield CPU to application, or EWA_STATUS_ERROR on
 * failure.
 */
extern EwaStatus ewaNetHTTPSend
  ( EwaNetHandle net_handle, EwaNetBuffer buffer );
#endif

#ifndef ewaNetHTTPEnd
/*
 * ewaNetHTTPEnd (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to indicate the completion of a request after all response
 * data has been sent.
 *
 * net_handle   - application-specific request handle from ewsNetHTTPStart()
 *
 * Returns EWA_STATUS_OK on success, EWA_STATUS_OK_YIELD on success and request
 * EmWeb/Server to yield CPU to application, or EWA_STATUS_ERROR on failure.
 */
extern EwaStatus ewaNetHTTPEnd ( EwaNetHandle net_handle );
#endif

#ifndef ewaNetBufferAlloc
/*
 * ewaNetBufferAlloc (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to request a network buffer to be used for sending data.
 *
 * Returns a buffer, or EWA_NET_BUFFER_NULL on failure. The length of the
 * buffer must be initialized to the number of bytes available for use
 * by EmWeb/Server.
 */
extern EwaNetBuffer ewaNetBufferAlloc ( void );
```

```
#endif

#ifndef ewaNetBufferFree
/*
 * ewaNetBufferFree (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to release one or a chain of network buffers (either from
 * ewsNetHTTPReceive() or ewaNetBufferAlloc()).
 *
 * buffer       - buffer(s) to be released
 *
 * No return value
 */
extern void ewaNetBufferFree ( EwaNetBuffer buffer );
#endif

#ifndef ewaNetBufferLengthGet
/*
 * ewaNetBufferLengthGet (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to get the length of the data portion of this buffer fragment.
 *
 * buffer       - buffer descriptor
 *
 * Returns length of buffer
 */
uintf ewaNetBufferLengthGet ( EwaNetBuffer buffer );
#endif

#ifndef ewaNetBufferLengthSet
/*
 * ewaNetBufferLengthSet (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to set the length of the data portion of this buffer fragment.
 * This function is only used to decrease the original length of a
 * buffer.  EmWeb/Server never increases the length of a buffer.
 *
 * buffer       - buffer descriptor
 * length       - new length value
 *
 * No return value
 */
void ewaNetBufferLengthSet ( EwaNetBuffer buffer, uintf length );
#endif

#ifndef ewaNetBufferDataGet
/*
 * ewaNetBufferDataGet (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to get the pointer to the data contained in the buffer.
 *
 * buffer       - buffer descriptor
 *
 * Returns pointer to data in buffer
```

```
 */
uint8 * ewaNetBufferDataGet ( EwaNetBuffer buffer );
#endif

#ifndef ewaNetBufferDataSet
/*
 * ewaNetBufferDataSet (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to set the pointer to the data contained in the buffer.
 * This function is only used to advance the start of data forward.
 * EmWeb/Server never moves this pointer backward.
 *
 * buffer       - buffer descriptor
 * datap        - new start of data value
 *
 * No return value
 */
void ewaNetBufferDataSet ( EwaNetBuffer buffer, uint8 *datap );
#endif

#ifndef ewaNetBufferNextGet
/*
 * ewaNetBufferNextGet (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to get the next buffer descriptor in the buffer chain.
 *
 * buffer       - buffer descriptor
 *
 * Returns next buffer descriptor, or EWA_NET_BUFFER_NULL if end of chain.
 */
EwaNetBuffer ewaNetBufferNextGet ( EwaNetBuffer buffer );
#endif

#ifndef ewaNetBufferNextSet
/*
 * ewaNetBufferNextSet (Application)
 * This function must be provided by the application.  It is invoked by
 * EmWeb/Server to set the next buffer descriptor in the buffer chain.
 *
 * buffer       - buffer descriptor
 * next         - next buffer descriptor to be attached to buffer
 *
 * No return value
 */
void ewaNetBufferNextSet ( EwaNetBuffer buffer, EwaNetBuffer next );
#endif

#ifndef ewaNetLocalHostName
/*
 * ewaNetLocalHostName (Application)
 * This function must be provided by the application.  It is invoked by the
 * EmWeb/Server to build proper redirection (Location:) headers.  This can
 * be either a dotted IP address or a fully qualified hostname.
 */
```

```
const char * ewaNetLocalHostName ( EwsContext context );
#endif

#ifndef ewaNetHTTPCleanup
/*
 * ewaNetHTTPCleanup
 * This function must be provided by the application or defined as a
 * empty macro.  It is invoked by EmWeb/Server when a request completes,
 * allowing the application to reset any processing state stored in the
 * network handle.  Note that for HTTP 1.1 persistent connections, this
 * routine may be called several times for the same connection, as one
 * connection can be used for multiple requests.  For the last request
 * on a connection, this routine will be called before ewaNetHTTPEnd is
 * invoked.
 *
 * Returns void.  Note that the request context is undefined during
 * this call and cannot be accessed.
 */
void ewaNetHTTPCleanup( EwaNetHandle handle );
#endif
#endif /* _EWS_NET_H */
```

A-2.5. src/include/ews_doc.h

This file contains the definitions and prototypes for the archive and document management functions of the application interface.

```
/*
 * ews_doc.h,v 1.20 1997/05/21 21:55:44 ian Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
 * http://www.agranat.com/
 *
 * EmWeb/Server application interface to document archives
 *
 */
```

```
#ifndef _EWS_DOC_H
#define _EWS_DOC_H

#include "ew_types.h"
#include "ew_config.h"
#include "ews_def.h"


/*****************************************************************************
 * ARCHIVE MAINTENANCE
 *
 * The EmWeb/Compiler generates an archive of one or more documents.  Documents
 * can be HTML files, JAVA programs, graphical images, or any other information
 * resource addressable by a URL.  Archives may be independently loaded or
 * unloaded into the EmWeb/Server.
 *
 * In most applications, the entire set of available documents would be
 * compiled into a single archive and loaded at boot time.  However, some
 * applications may desire to dynamically load archives at run-time as needed
 * in order to reduce memory requirements.  In fact, some applications may
 * want to implement a scheme similar to page swapping under some operating
 * systems to cache an active set of documents in memory while storing other
 * documents in a secondary storage area.  Such a secondary storage area
 * could be in FLASH memory, or on a remote server using TFTP or other
 * protocols to load documents at run-time.
 *
 * An EmWeb archive consists of two components.  First, there is the archive
 * data component containing the database of compressed documents, information
 * about how to construct dynamic documents at run-time, access controls,
 * etc.  Second, there is the archive object component containing the run-time
 * object code used for the construction of dynamic documents, etc.
 *
 * Operating systems supporting the run-time loading and linking of object
 * code may offload both the data and object archive components to a secondary
 * storage area.  Otherwise, only the data components would be offloaded
 * while the object components would be statically linked into the run-time
 * executable image.
 *
 * Each archive contains an archive descriptor in the object component.  The
 * archive descriptor is referenced by a public symbol generated by the
 * EmWeb/Compiler.  In order to activate an archive at run-time, the
 * application must invoke ewsDocumentInstallArchive() with parameters
 * indicating the address of the data component and the archive descriptor of
 * the object component.  The archive may be deactivated by invoking
 * ewsDocumentRemoveArchive().
 *
 *****************************************************************************/

/*
 * ewsDocumentInstallArchive
 *
 * Install a document archive generated by EmWeb/Compiler into EmWeb/Server
 * run-time database.
 *
```

```
 * descriptor    - public symbol of archive object component descriptor
 * datap         - pointer to archive data component
 *
 * Returns EWS_STATUS_OK on success, else error code (TBD)
 */
extern EwsStatus ewsDocumentInstallArchive
  ( EwsArchive descriptor, EwsArchiveData datap );

#ifdef EW_CONFIG_OPTION_CLEANUP

/*
 * ewsDocumentRemoveArchive
 *
 * Removes previously installed document archive from run-time database.
 *
 * descriptor    - public symbol of archive object component descriptor
 *
 * Returns EWS_STATUS_OK on success, else error code (TBD)
 */
extern EwsStatus ewsDocumentRemoveArchive ( EwsArchive descriptor );

#endif /* EW_CONFIG_OPTION_CLEANUP */

/*
 * ewsDocumentArchiveName
 *
 * Returns the archive name as stored in the archive by the EmWeb/Compiler,
 * or NULL on error.
 */
extern const char * ewsDocumentArchiveName ( const uint8 *datap );

/*
 * ewsDocumentArchiveDate
 *
 * Returns the RFC1123 date string stored in the archive header generated
 * by the EmWeb/Compiler when the archive was created, or NULL on error.
 *
 * There is also a routine for the less preferred RFC1036 representation.
 */
extern const char * ewsDocumentArchiveDate ( const uint8 *datap );
extern const char * ewsDocumentArchiveDate1036 ( const uint8 *datap );

#ifdef EW_CONFIG_OPTION_DEMAND_LOADING

/*****************************************************************************
 *
 * DEMAND LOADING
 *
 * In order to implement on-demand archive loading, the application may
 * register document URLs with the EmWeb/Server which are valid but not
 * loaded.  This is done by invoking ewsDocumentRegister().  If a registered
 * document is requested, the EmWeb/Server will notify the application by
 * invoking ewaDocumentFault().  At this point, the application may load
 * a new archive (possibly removing a previously installed archive to make
```

```
 * room).  When the archive containing the page is installed, the EmWeb/Server
 * will automatically complete processing the request.  The request can be
 * aborted either immediately by returning EWA_STATUS_ERROR from the
 * ewaDocumentFault() function, or by invoking ewsNetHTTPAbort().
 *
 * Once a document is registered, there is no need to re-register it in the
 * event that the corresponding archive is removed.  EmWeb/Server remembers
 * that the document has been registered as dynamically loadable.  However,
 * the application may deregister a document by invoking
 * ewsDocumentDeregister().
 *
 ************************************************************************/

/*
 * ewsDocumentRegister
 * Registers the URL of a document with the run-time database indicating that
 * the given document is valid but not currently loaded into memory.
 *
 * url          - local URL of document to be registered
 * handle       - application-specific handle passed to ewaDocumentFault()
 *
 * Returns document descriptor, or EWS_DOCUMENT_NULL on failure.
 */
extern EwsDocument ewsDocumentRegister
  ( const char * url, EwaDocumentHandle handle );

/*
 * ewsDocumentDeregister
 * Unregisters a previously registered document from the run-time database
 * indicating that the given dynamically-loadable document is no longer valid.
 *
 * document     - descriptor of document to be deregistered
 *
 * Returns EWS_STATUS_OK on success, else error code (TBD).
 */
extern EwsStatus ewsDocumentDeregister ( EwsDocument document );

#ifndef ewaDocumentFault
/*
 * ewaDocumentFault (Application)
 * This function must be provided by the application, and is invoked by
 * EmWeb/Server when a registered document has been requested but is not
 * currently loaded into memory.
 *
 * context      - context of request for document
 * handle       - application-specific handle from ewsDocumentRegister()
 *
 * Returns EWA_STATUS_OK on success, EWA_STATUS_OK_YIELD on success with
 * request for EmWeb/Server to yield the CPU, or EWA_STATUS_ERROR on failure.
 * If failure status is returned, the request is aborted.  Otherwise, the
 * request will complete after the document has been loaded into memory.
 */
extern EwaStatus ewaDocumentFault
  ( EwsContext context, EwaDocumentHandle handle );
```

Copyright © 1997 Agranat Systems, Inc.

```
#endif

#endif /* EW_CONFIG_OPTION_DEMAND_LOADING */

#ifdef EW_CONFIG_OPTION_CLONING

/*****************************************************************************
 *
 * DOCUMENT CLONING
 *
 * Documents may be cloned and assigned to a new URL.  This allows multiple
 * instances of a document to exist.  An application-specific handle can be
 * used to identify an instance of a document from the request context.
 *
 * The application clones a document by invoking ewsDocumentClone().  The
 * cloned document may be removed by invoking ewsDocumentRemove().  All
 * clones created from documents in an archive must be removed before the
 * archive can be removed.
 *
 *****************************************************************************/

/*
 * ewsDocumentClone
 * Clone an existing document under a new URL in the database.
 *
 * baseurl      - URL of existing document
 * newurl       - URL of new cloned document
 * handle       - application-specific handle available in request context
 *
 * Returns document descriptor or EWS_DOCUMENT_NULL on error.
 */
extern EwsDocument ewsDocumentClone
  ( const char * baseurl, const char * newurl, EwaDocumentHandle handle );

/*
 * ewsDocumentRemove
 * Remove a previously cloned document from the database.
 *
 * document     - document descriptor of previously cloned document.
 *
 * Returns EWS_STATUS_OK on success, else error code (TBD).
 */
extern EwsStatus ewsDocumentRemove ( EwsDocument document );
#endif /* EW_CONFIG_OPTION_CLONING */

#ifdef EW_CONFIG_OPTION_DOCUMENT_DATA
/*
 * ewsDocumentData
 * Retrieve pointer to raw data and data size from URL in archive
 *
 * url          - url of document
 * bytesp       - output: size of document (as-is in archive) in bytes
 * datapp       - output: pointer to start of document
 *
```

```
 * Returns EWS_STATUS_OK on success, else error code (TBD).
 */
extern EwsStatus ewsDocumentData ( const char * url
                                  ,uint32 *bytesp
                                  ,const uint8 **datapp );
#endif /* EW_CONFIG_OPTION_DOCUMENT_DATA */

#ifdef EW_CONFIG_OPTION_DOCUMENT_SET_REALM
#ifdef EW_CONFIG_OPTION_AUTH
/*
 * ewsDocumentSetRealm
 * Set or change authentication realm of document at run-time.
 *
 * url          - url of document
 * realm        - name of authentication realm, or NULL (or "") to disable.
 *
 * Returns EWS_STATUS_OK on success, else error code.
 */
extern EwsStatus ewsDocumentSetRealm ( const char * url, const char *realm );
#endif /* EW_CONFIG_OPTION_AUTH */
#endif /* EW_CONFIG_OPTION_DOCUMENT_SET_REALM */

#endif /* _EWS_DOC_H */
```

A-2.6. src/include/ews_auth.h

This file contains the definitions and prototypes of the authorization and security functions of the application interface.

```
/*
 * ews_auth.h,v 1.18 1997/05/21 21:55:44 ian Exp
 *
 * Product:     EmWeb
 * Release:     R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
 * http://www.agranat.com/
 *
 * EmWeb/Server application interface to authorization database
```

```
 *
 */

#ifndef _EWS_AUTH_H_
#define _EWS_AUTH_H_

#ifdef EW_CONFIG_OPTION_AUTH

#include "ew_types.h"
#include "ew_config.h"

/*****************************************************************************
 *
 * HTTP Authorization techniques are discussed in the HTTP specifications.
 * Knowledge of these principles are essential for correct use of these
 * authorization interfaces.
 *
 * One or more HTML documents can be assocated with a single "realm".  A realm
 * is a case-sensitive ascii string that defines a "protection space" for the
 * associated document(s).
 *
 * Each realm can have zero or more authentication entries associated with it.
 * If a realm has no authentication entries, then it is considered
 * "unprotected" by the server, and its acess will always be granted to any
 * requesting client.
 *
 * For those realms with authentication entries, any client that attempts to
 * access any document associated with that realm will be required to
 * authenticate itself.  To authenticate, the client must send authentication
 * parameters that validate against at least one authentication entry for that
 * document's realm.  Clients that do not authenticate are not served the
 * requested document.
 *
 * For example, assume a realm exists called "foo".  It has three
 * authentication entries associated with it (2 using the "basic" scheme
 * defined in the HTTP specification, and one other ficticious type):
 *
 * REALM: "foo"
 *         Authentication Entry 1:
 *              Type: "basic"
 *              parameters: Username="user1"
 *                          Password="guest"
 *              EwaAuthHandle:  <application's entry1 identifier>
 *
 *         Authentication Entry 2:
 *              Type: "future"
 *              parameters: index=37
 *                          cookie="837I8U9"
 *                          token="S37"
 *              EwaAuthHandle:  <application's entry2 identifier>
 *
 *         Authentication Entry 3:
 *              Type: "basic"
 *              parameters: Username="sinclair"
```

```
 *                      Password="babylon"
 *             EwaAuthHandle:  <application's entry3 identifier>
 *
 * When a client attempts to access a document associated with realm "foo", it
 * will need to authenticate against one of the above entries.  Which one the
 * client authenticates against is at its disgression.
 *
 * When a client does authenticate against one of the above entries, the
 * EwaAuthHandle for that entry is stored in the current context for the
 * document (EwsContext).  The datatype of this EwaAuthHandle is
 * implementation-defined.
 ****************************************************************************/

/* ------------------------------------------------------------
 * Types & Constants
 * ------------------------------------------------------------ */

/*
 * Defines the authorization schemes supported by the EmWeb server.
 * New schemes will be added as they are supported.
 */
typedef enum
{

#   ifdef EW_CONFIG_OPTION_AUTH_BASIC
    ewsAuthSchemeBasic,
#   endif /* EW_CONFIG_OPTION_AUTH_BASIC */

#   ifdef EW_CONFIG_OPTION_AUTH_DIGEST
    ewsAuthSchemeDigest,
#   endif /* EW_CONFIG_OPTION_AUTH_DIGEST */

#   ifdef EW_CONFIG_OPTION_AUTH_MBASIC
    ewsAuthSchemeManualBasic,
#   endif /* EW_CONFIG_OPTION_AUTH_MBASIC */

#   ifdef EW_CONFIG_OPTION_AUTH_MDIGEST
    ewsAuthSchemeManualDigest,
#   endif /* EW_CONFIG_OPTION_AUTH_MDIGEST */

    ewsAuthMaxScheme            /* count of supported schemes */

} EwsAuthScheme;


/*
 * this structure defines parameters for all the supported
 * authorization types.  New parameters will be added in
 * to this structure in the future
 */
typedef union
{
#ifdef EW_CONFIG_OPTION_AUTH_BASIC
  struct
```

```
  {
        char   *userName;
        char   *passWord;
  } basic;
#endif /* EW_CONFIG_OPTION_AUTH_BASIC */
#if defined(EW_CONFIG_OPTION_AUTH_DIGEST) ||
defined(EW_CONFIG_OPTION_AUTH_MDIGEST)
  struct
  {
        char     *userName;
        char     *passWord;

#       ifdef EW_CONFIG_OPTION_AUTH_DIGEST_M
        boolean digestRequired; /* require client message data verification */
#       endif /* EW_CONFIG_OPTION_AUTH_DIGEST_M */

  } digest;
#endif /* EW_CONFIG_OPTION_AUTH_DIGEST || EW_CONFIG_OPTION_AUTH_MDIGEST */
  char reserved1;
} EwsAuthParameters;


/*
 * This structure represents a single authorization entry.
 */
typedef struct
{
  EwsAuthScheme     scheme;
  EwsAuthParameters params;
  EwaAuthHandle     handle;     /* user defined */
} EwsAuthorization;


/* ------------------------------------------------------------
 * Interfaces
 * ------------------------------------------------------------ */


/*
 * ewsAuthRegister
 * And a new authorization entry to the realm.
 *
 * realm - Input, case sensitive asciiz realm name.
 * authorization - Input, buffer containing authorization entry information.
 *                 This buffer is free for use by the caller on return.
 *
 * Returns a non-NULL handle to the authorization entry, or NULL on failure.
 */

extern EwsAuthHandle ewsAuthRegister
  ( const char *realm,const EwsAuthorization *authorization );


/*
```

```
 * ewsAuthRemove
 * Deletes a particular authentication entry identified by handle.
 *
 * handle - Input, The handle that was returned by ewsAuthRegister() for
 *          this entry.
 *
 * Returns EWS_STATUS_OK on success, else failure code.
 */
extern EwsStatus ewsAuthRemove ( EwsAuthHandle handle );

/*
 * ewsAuthRealmEnable
 * Enables a realm to restrict access to documents (restricted by default)
 *
 * realm - Input, the realm to restrict
 *
 * Returns EWS_STATUS_OK on success, else failure code
 */
extern EwsStatus ewsAuthRealmEnable ( const char * realm );

/*
 * ewsAuthRealmDisable
 * Disables a realm causing documents in that realm to be generally accessible.
 *
 * realm - Input, the realm to restrict
 *
 * Returns EWS_STATUS_OK on success, else failure code
 */
extern EwsStatus ewsAuthRealmDisable ( const char * realm );

/*
 * ewaAuthRealmQualifier
 * Returns realm qualifier string to append to realm names, or NULL
 */
#ifndef ewaAuthRealmQualifier
extern const char *ewaAuthRealmQualifier ( void );
#endif /* ewaAuthRealmQualifier */

#if defined(EW_CONFIG_OPTION_AUTH_DIGEST) \
 || defined(EW_CONFIG_OPTION_AUTH_MDIGEST)
/*
 * ewaAuthNonceCreate
 * Create raw input used to generate one-time authentication challenges
 * given a request context and realm.
 *
 * context - request context
 * realm   - null-terminated string containing name of realm
 * noncep  - pointer to application-specific nonce parameters
 *
 * No return value
 */
extern void ewaAuthNonceCreate
  ( EwsContext context, const char * realm, EwaAuthNonce *noncep );
```

```
/*
 * ewaAuthNonceCheck
 * Validate that the previously created nonce is valid given the current
 * request.
 *
 * context - request context
 * realm   - null-terminated string containing name of realm
 * noncep  - pointer to requested nonce parameters
 * count   - previous use count (0, 1, ...) of nonce value
 *
 * Returns ewaAuthNonceOK on success, ewaAuthNonceStale if valid but expired,
 * or ewaAuthNonceDenied if nonce is not permitted given the request context.
 */
typedef enum EwaAuthNonceStatus_e
  {
    ewaAuthNonceOK,     /* nonce value valid for request */
    ewaAuthNonceLastOK, /* nonce value valid, but won't be again */
    ewaAuthNonceStale,  /* nonce value is stale, generate new nonce */
    ewaAuthNonceDenied  /* nonce value is invalid */
  } EwaAuthNonceStatus;

extern EwaAuthNonceStatus ewaAuthNonceCheck
  ( EwsContext context, const char * realm, EwaAuthNonce *noncep, uintf count );
#endif /* EW_CONFIG_OPTION_AUTH_(M)DIGEST */


#ifdef EW_CONFIG_OPTION_AUTH_MBASIC
/*
 * ewaAuthCheckBasic
 * Given the current context, realm, base64 cookie, and optionally
 * decoded username/password text, determine if the authorization should
 * be granted or denied.
 *
 * context - (input) request context
 * realm   - (input) null-terminated string containing name of realm
 * basicCookie - (input) the base64 encoding of the text user:password,
 *           as described in the HTTP RFC.
 * userName - (input) if EW_CONFIG_OPTION_AUTH_MBASIC_DECODE is defined,
 *          then this parameter points to a null terminated string
 *          containing the user name decoded from the basicCookie.  If
 *          EW_CONFIG_OPTION_AUTH_MBASIC_DECODE is not defined, this
 *          is a NULL pointer.
 * password - (input) if EW_CONFIG_OPTION_AUTH_MBASIC_DECODE is defined,
 *          then this parameter points to a null terminated string
 *          containing the password decoded from the basicCookie.  If
 *          EW_CONFIG_OPTION_AUTH_MBASIC_DECODE is not defined, this
 *          is a NULL pointer.
 *
 * Return value: boolean, if TRUE, then the authorization is granted and
 * the document is served.  Otherwise, authorization is denied.
 *
 * This function can be supended by ewsSuspend().  It's return values
 * are ignored, and it will be recalled with the same parameters when
 * the context is resumed using ewsResume().
```

```
 */

#ifndef ewaAuthCheckBasic
extern boolean ewaAuthCheckBasic( EwsContext context
                                 ,const char *realm
                                 ,const char *basicCookie
                                 ,const char *userName
                                 ,const char *password );
#endif /* ewaAuthCheckBasic */
#endif /* EW_CONFIG_OPTION_AUTH_MBASIC */


#ifdef EW_CONFIG_OPTION_AUTH_MDIGEST
/*
 * ewaAuthCheckDigest
 * Given the current context, realm, username, and nonce,
 * return the corresponding MD5 hash of the username:realm:password
 * text.  This text will be used by the server to verify the
 * authorization attempt by the client.
 *
 * context - (input) request context
 * realm   - (input) null-terminated string containing name of realm
 *           taken from the request header.
 * nonce   - (input) null-terminated string containing the nonce that
 *           was supplied to the server.
 * userName - (input) this parameter points to a null terminated string
 *           containing the user name as supplied in the authorization
 *           request header.
 * digest  - (output) on return, *digest will point to a null-terminated
 *           ascii text string representing the MD5 checksum of the
 *           username, realm and password (MD5( username:realm:password))
 *           as described in the Digest Access Authentication RFC2069
 *
 * Return value: boolean, if TRUE, then *digest points to a valid MD5 hash of
 * username:realm:password, which will be used by the server to complete
 * the authorization process (which may or may not be granted).  If FALSE,
 * the authorization is immediately denied.
 *
 * This function can be supended by ewsSuspend().  It's return values
 * are ignored, and it will be recalled with the same parameters when
 * the context is resumed using ewsResume().
 */

#ifndef ewaAuthCheckDigest
extern boolean ewaAuthCheckDigest( EwsContext context
                                  ,const char *realm
                                  ,const EwaAuthNonce *noncep
                                  ,const char *userName
                                  ,const char **digest );
#endif /* ewaAuthCheckDigest */
#endif /* EW_CONFIG_OPTION_AUTH_MDIGEST */


#ifdef EW_CONFIG_OPTION_AUTH_VERIFY
```

```
/*
 * ewaAuthVerifySecurity
 * Called _after_ the server has determined that the authorization
 * will be granted for the request, this function allows the application
 * to "short circuit" the authorization based on information about
 * the request (eg, IP address of client, etc.).  The method used
 * to determine whether or not the request is authorized is proprietary
 * to the application.
 *
 * context - (input) request context
 * realm   - (input) null-terminated string containing name of realm
 *           taken from the request header.
 *
 * Return value: boolean, if TRUE, then the authorization is granted, and
 * the document will be served to the client.  If FALSE,
 * the authorization is immediately denied.
 *
 * This function can be supended by ewsSuspend().  It's return values
 * are ignored, and it will be recalled with the same parameters when
 * the context is resumed using ewsResume().
 */

#ifndef ewaAuthVerifySecurity
extern boolean ewaAuthVerifySecurity( EwsContext context
                                     ,const char *realm );
#endif /* ewaAuthVerifySecurity */
#endif /* EW_CONFIG_OPTION_AUTH_VERIFY */


/*
 *** Local Variables: ***
 *** mode: c ***
 *** tab-width: 4 ***
 *** End: ***
 */

#endif /* EW_CONFIG_OPTION_AUTH */

#endif /* _EWS_AUTH_H_ */
```

A-2.7. src/include/ews_ctxt.h

This file contains definitions and prototypes of the request context access functions of the application interface.

```
/*
 * ews_ctxt.h,v 1.14 1997/05/25 23:30:32 giusti Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
```

```
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
 * http://www.agranat.com/
 *
 * EmWeb/Server public definitions
 *
 */
#ifndef _EWS_CONTEXT_H
#define _EWS_CONTEXT_H

#include "ew_types.h"
#include "ew_config.h"
#include "ews_def.h"

/*
 * REQUEST CONTEXT
 *
 * Each HTTP request received by EmWeb/Server is assigned a unique context.
 * The context structure contains all the information pertaining to the
 * request, as well as internal state information regarding the processing
 * state of the request.
 *
 * API functions are provided to give the application some visibility into
 * the current context state.
 */

/*
 * ewsContextNetHandle
 * Return the application-specific network handle provided to EmWeb/Server
 * by the application in ewsNetHTTPStart().
 */
EwaNetHandle ewsContextNetHandle ( EwsContext context );

#if defined (EW_CONFIG_OPTION_DEMAND_LOADING) \
    || defined (EW_CONFIG_OPTION_CLONING)
/*
 * ewsContextDocumentHandle
 * Return the application-specific document handle provided to EmWeb/Server
 * by the application in ewsDocumentRegister() or ewsDocumentClone().
 */
EwaDocumentHandle ewsContextDocumentHandle ( EwsContext context );
#endif /* EW_CONFIG_OPTION_DEMAND_LOADING || EW_CONFIG_OPTION_CLONING */
```

```
#ifdef EW_CONFIG_OPTION_AUTH
/*
 * ewsContextAuthHandle
 * Return the application-specific authorization handle provided to
 * EmWeb/Server by the application in ewsAuthRegister().
 */
EwaAuthHandle ewsContextAuthHandle ( EwsContext context );
#endif /* EW_CONFIG_OPTION_AUTH */

#ifdef EW_CONFIG_OPTION_SCHED_SUSP_RES
/*
 * ewsContextIsResuming
 * Return TRUE if EmWeb/Server is resuming (as a result of ewsResume()) after
 * the context was suspended (as a result of ewsSuspend()).
 */
boolean ewsContextIsResuming ( EwsContext context );
#endif /* EW_CONFIG_OPTION_SCHED_SUSP_RES */

#if defined (EW_CONFIG_OPTION_STRING) || defined (EW_CONFIG_OPTION_INCLUDE)
/*
 * ewsContextIterations
 * Returns the iteration count corresponding for EMWEB_ITERATE
 */
uint32 ewsContextIterations ( EwsContext context );
#endif /* EW_CONFIG_OPTION_STRING || EW_CONFIG_OPTION_INCLUDE */

#ifdef EW_CONFIG_OPTION_CONTEXT_SEND_REPLY
/*
 * ewsContextSendReply
 * Serves specified local path URL to browser in response to form submission
 * or raw CGI.
 */
EwsStatus ewsContextSendReply ( EwsContext context, char * url );
#endif /* EW_CONFIG_OPTION_CONTEXT_SEND_REPLY */

#if defined( EW_CONFIG_OPTION_FILE_GET ) \
 || defined( EW_CONFIG_OPTION_FILE_PUT ) \
 || defined( EW_CONFIG_OPTION_FILE_DELETE )
EwsStatus ewsContextSetFile( EwsContext context
                            ,EwsFileParamsP params
                           );
#endif /* EW_CONFIG_OPTION_FILE_xxx */


/*
 * The following functions extract the corresponding fields from the HTTP/1.0
 * request header, if present, into an application-provided buffer.  Each of
 * these functions return the number of bytes in the actual header value, even
 * if this is larger than the application-provided buffer (though EmWeb/Server
 * will not overwrite the applicatoin buffer by honoring the buffer length).
 * If the header is not present, zero is returned.  The application may
 * specify a zero-length buffer in order to determine the size of the header
 * value.  The returned value begins with the first non-whitespace following
 * the HTTP header and ends just before the terminating end-of-line
```

```
 * character(s).
 */
#ifdef EW_CONFIG_OPTION_CONTEXT_DATE
uintf ewsContextDate ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CONTEXT_DATE */
#ifdef EW_CONFIG_OPTION_CONTEXT_PRAGMA
uintf ewsContextPragma ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CONTEXT_PRAGMA */
#ifdef EW_CONFIG_OPTION_CONTEXT_FROM
uintf ewsContextFrom ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CONTEXT_FROM */
#ifdef EW_CONFIG_OPTION_CONTEXT_IF_MODIFIED_SINCE
uintf ewsContextIfModifiedSince
  ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CONTEXT_IF_MODIFIED_SINCE */
#ifdef EW_CONFIG_OPTION_CONTEXT_REFERER
uintf ewsContextReferer ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CONTEXT_REFERER */
#ifdef EW_CONFIG_OPTION_CONTEXT_USER_AGENT
uintf ewsContextUserAgent ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CONTEXT_USER_AGENT */
#ifdef EW_CONFIG_OPTION_CONTEXT_HOST
uintf ewsContextHost ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CONTEXT_HOST */

#endif /* _EWS_CONTEXT_H */
```

A-2.8. src/include/ews_cgi.h

This file contains the definitions and prototypes of the raw CGI functions of the application interface.

```
/*
 * ews_cgi.h,v 1.22 1997/03/29 17:50:49 ian Exp
 *
 * Product: EmWeb
 * Release: R2_1
 *
 * CONFIDENTIAL AND PROPRIETARY INFORMATION OF AGRANAT SYSTEMS, INC.
 * THE EMWEB SOFTWARE ARCHITECTURE IS PATENT PENDING.
 * EMWEB IS A TRADEMARK OF AGRANAT SYSTEMS, INC.
 *
 * Copyright (C) 1996, 1997 Agranat Systems, Inc.
 * All Rights Reserved
 *
 * Agranat Systems, Inc.
 * 1345 Main Street
 * Waltham, MA  02154
 *
 * (617) 893-7868
 * sales@agranat.com, support@agranat.com
 *
```

```
 * http://www.agranat.com/
 *
 * EmWeb/Server application interface to raw CGI
 *
 */
#ifndef _EWS_CGI_H
#define _EWS_CGI_H

#include "ew_types.h"

/*****************************************************************************
 * CGI INTERFACE
 *
 * EmWeb supports a raw CGI interface for imagemap and various special
 * application-defined CGI capabilities.
 *
 * The EmWeb/Compiler creates CGI URLs from specifications provided in
 * _ACCESS files.  The application provides two functions for each CGI URL.
 * First, there is the CGI start function.  This is invoked by the
 * EmWeb/Server to mark the start of a new CGI request.  Second, there is the
 * CGI data function.  This is invoked by the EmWeb/Server one or more times
 * to pass network buffers containing raw CGI request data to the application.
 * The end of request data is indicated by an EWA_NET_BUFFER_NULL EwaNetBuffer.
 *
 * The CGI application uses the ewsCGIData() function to transmit response data
 * to the network.
 *
 *****************************************************************************/

/*
 * ewaCGIStart
 *
 * This per-CGI function is provided by the application.  It is invoked by
 * EmWeb/Server at the start of a new request to the corresponding CGI URL.
 * The application may return an application-specific handle.
 *
 * context      - context of current request
 *
 * Returns application-specific handle to be used by EmWeb/Server in
 * subsequent calls to ewaCGIData.
 */
typedef EwaCGIHandle EwaCGIStart_f ( EwsContext context );

/*
 * ewaCGIData
 *
 * This per-CGI function is provided by the application.  It is invoked by
 * EmWeb/Server zero or more times to pass raw CGI request data to the
 * application.
 *
 * cgi_handle   - application-specific handle as defined by ewaCGIStart
 * buffer       - application buffer containing all or part of the raw CGI
 *                request data
 *
```

```
 * No return value
 */
typedef void EwaCGIData_f ( EwaCGIHandle cgi_handle, EwaNetBuffer buffer );

#ifdef EW_CONFIG_OPTION_CGI

/*
 * ewsCGISendStatus
 * This optional call causes the EmWeb/Server to construct and send an
 * appropriate  HTTP status header (and possibly other server-generated
 * headers).  It must be followed by one or more calls to ewsCGIData() to send
 * additional header information and optional body content.  (Note that a call
 * to ewsCGIData() must not precede the invocation of this function).
 *
 * Note: If the application desires complete control of the response, it may
 * simply send all header information itself using ewsCGIData() instead of
 * invoking this function.
 *
 * context      - context of request
 * status       - 3-digit status code followed by reason string
 * string       - string containing additional headers and/or data, or NULL
 *
 * Returns EWS_STATUS_OK on success, else error code (TBD).
 */
extern EwsStatus ewsCGISendStatus
  ( EwsContext context, const char * status, const char * string );

/*
 * ewsCGIData
 *
 * This function is called by a raw CGI application one or more times to send
 * raw CGI response data to the browser client.  The application is responsible
 * for throttling its use of the CPU.  Note that response data may include raw
 * HTTP headers as well as data.  Therefore, a CR-NL sequence must be
 * transmitted to properly delimit the header portion of the response from
 * any returned data.
 *
 * Note that ewsCGISendStatus() may be called once before any calls to
 * ewsCGIData() to generate standard HTTP headers.  Otherwise, the application
 * must generate all standard headers itself.
 *
 * Note that sending a EWA_NET_BUFFER_NULL terminates the request.
 *
 * context      - context of request
 * buffer       - buffer containing data to send, or EWA_NET_BUFFER_NULL
 *                if this is the last buffer in the response.
 *
 * No return value
 */
extern EwsStatus ewsCGIData ( EwsContext context, EwaNetBuffer buffer );

/*
 * ewsCGIRedirect
 * This call causes the EmWeb/Server to respond to the browser with the
```

```
 * document specified by the URL (if local), or with a redirect (if not).
 * This function will terminate the request.  Note that this feature can not
 * be used in conjunction with ewsCGIData() or ewsCGIStatus().
 *
 * context      - context of request
 * url          - URL for redirect
 *
 * Returns EWS_STATUS_OK on success, else error code (TBD).
 */
extern EwsStatus ewsCGIRedirect ( EwsContext context, const char *url );

#ifdef EW_CONFIG_OPTION_CGI_SERVER_SOFTWARE
/*
 * ewsCGIServerSoftware
 *
 * Returns a string corresponding to the server software such as
 * "Agranat-EmWeb/R2_1"
 */
extern const char *ewsCGIServerSoftware ( void );
#endif /* EW_CONFIG_OPTION_CGI_SERVER_SOFTWARE */

#ifdef EW_CONFIG_OPTION_CGI_GATEWAY_INTERFACE
/*
 * ewsCGIGatewayInterface
 *
 * Returns a string corresponding to the CGI version "CGI/1.0"
 */
extern const char *ewsCGIGatewayInterface ( void );
#endif /* EW_CONFIG_OPTION_CGI_GATEWAY_INTERFACE */

#ifdef EW_CONFIG_OPTION_CGI_SERVER_PROTOCOL
/*
 * ewsCGIServerProtocol
 *
 * context      - context of request
 * datap        - pointer to data buffer
 * length       - size of data buffer (may be zero)
 *
 * Returns number of bytes in actual protocol string, or zero if not present.
 */
extern uintf ewsCGIServerProtocol
  ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CGI_SERVER_PROTOCOL */

#ifdef EW_CONFIG_OPTION_CGI_REQUEST_METHOD
/*
 * ewsCGIRequestMethod
 *
 * context      - context of request
 * datap        - pointer to data buffer
 * length       - size of data buffer (may be zero)
 *
 * Returns number of bytes in actual request method, or zero if not present.
 */
```

```
extern uintf ewsCGIRequestMethod
  ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CGI_REQUEST_METHOD */

#ifdef EW_CONFIG_OPTION_CGI_PATH_INFO
/*
 * ewsCGIPathInfo
 *
 * context      - context of request
 *
 * Returns pointer to PathInfo string, or NULL if not present
 */
extern const char * ewsCGIPathInfo ( EwsContext context );
#endif /* EW_CONFIG_OPTION_CGI_PATH_INFO */

#ifdef EW_CONFIG_OPTION_CGI_SCRIPT_NAME
/*
 * ewsCGIScriptName
 *
 * context      - context of request
 *
 * Returns pointer to script name string
 */
extern const char * ewsCGIScriptName ( EwsContext context );
#endif /* EW_CONFIG_OPTION_CGI_SCRIPT_NAME */

#ifdef EW_CONFIG_OPTION_CGI_QUERY_STRING
/*
 * ewsCGIQueryString
 *
 * context      - context of request
 * datap        - pointer to data buffer
 * length       - size of data buffer (may be zero)
 *
 * Returns number of bytes in actual query string, or zero if not present.
 */
extern uintf ewsCGIQueryString
  ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CGI_QUERY_STRING */

#ifdef EW_CONFIG_OPTION_CGI_CONTENT_TYPE
/*
 * ewsCGIContentType
 *
 * context      - context of request
 * datap        - pointer to data buffer
 * length       - size of data buffer (may be zero)
 *
 * Returns number of bytes in actual content type, or zero if not present.
 */
extern uintf ewsCGIContentType
  ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CGI_CONTENT_TYPE */
```

```
#ifdef EW_CONFIG_OPTION_CGI_CONTENT_LENGTH
/*
 * ewsCGIContentLength
 *
 * context      - context of request
 *
 * Returns content length as specified by client, or zero if not present,
 * or EWS_CGI_CONTENT_LENGTH_CHUNKED if chunked.
 */
#define EWS_CGI_CONTENT_LENGTH_CHUNKED  0xffffffff
extern uint32 ewsCGIContentLength ( EwsContext context );
#endif /* EW_CONFIG_OPTION_CGI_CONTENT_LENGTH */

#ifdef EW_CONFIG_OPTION_CGI_CONTENT_ENCODING
/*
 * ewsCGIContentEncoding
 *
 * context      - context of request
 * datap        - pointer to data buffer
 * length       - size of data buffer (may be zero)
 *
 * Returns number of bytes in actual content encoding, or zero if not present.
 */
extern uintf ewsCGIContentEncoding
  ( EwsContext context, char *datap, uintf length );
#endif /* EW_CONFIG_OPTION_CGI_CONTENT_ENCODING */

#endif /* EW_CONFIG_OPTION_CGI */

#endif /* _EWS_CGI_H */
```

TABLE B

```
ew_db.h        Sat Jun 28 07:59:50 1997        1
/*
 * $Id: ew_db.h,v 1.30 1997/06/28 11:59:50 ian Exp $
 *
 * Product: EmWeb
 * Release: $Name:  $
 *
 * %CopyRight%
 *
 * EmWeb database definitions
 *
 */
#ifndef _EW_DB_H
#define _EW_DB_H

#include "ew_common.h"
#include "ew_types.h"
#include "ew_form.h"
#include "ew_imap.h"
#include "ews_def.h"
#include "ews_cgi.h"

/*****************************************************************************
 *
 * DOCUMENT ARCHIVE
 *
 * An EmWeb document archive contains two parts.  These are the archive data
 * component and the archive object component.
 *
 * The archive object component contains all the run-time object code
 * associated with an archive.  A public symbol (determined by EmWeb/Compiler
 * on a per-archive basis) points to an archive descriptor (EwsArchive).  This
 * contains all of the linkage required by the EmWeb/Server to access the
 * archive-specific run-time object code.
 *
 * The archive data component contains all the static data describing
 * individual documents in the archive.  The data component is an endian-
 * independent fully relocatable contiguous block of constant data
 * analogous to a file.  In fact, it could be a file.  The data component
 * begins with an archive header and contains one or more documents.
 *****************************************************************************/

/*
 * Archive Descriptor
 *
 * Each archive has a unique archive descriptor in the object component of the
 * archive.  The archive descriptor is a list of functions called by the
 * EmWeb/Server to execute archive-specific object code while processing
 * requests for documents in the archive.  Unsupported functions (i.e.
 * features not selected when the EmWeb/Compiler was built)  are represented
 * by a NULL pointer.
 */

typedef const void * EmWebString_f     ( EwsContext context, uint32 index );
typedef const char * EmWebInclude_f    ( EwsContext context, uint32 index );

/* Cookie Attribute structure, element of emweb_cookie_tbl[]            */
typedef struct EwCookieAttr_s {
    char *name;
    char *path;
} EwCookieAttributes;


typedef struct EmWebCGITable_s
  {
    EwaCGIStart_f      *start_f;
```

```
ew_db.h        Sat Jun 28 07:59:50 1997        2
    EwaCGIData_f      *data_f;
  } EmWebCGITable, *EmWebCGITableP;

typedef struct EwsArchive_s
  {
    uint32      magic;          /* Magic number for archive validation */

#   define EW_ARCHIVE_OBJECT_MAGIC      (('E'<<24)|('W'<<16)|('o'<<8)|'b')

    uint8       version_maj;    /* Archive version (major/minor) */
    uint8       version_min;

#   define EW_ARCHIVE_VERSION_MAJ       1
#   define EW_ARCHIVE_VERSION_MIN       1

    uint8       compiler_maj;   /* EmWeb/Compiler version (major/minor) */
    uint8       compiler_min;

#   define EW_ARCHIVE_DATE_1123_SIZE    32

    char        date_1123[EW_ARCHIVE_DATE_1123_SIZE];   /* creation date */

    /*
     * Archive-specific <EMWEB_STRING> dispatch function.  Takes context
     * and index arguments.  Index selects the C-code fragment corresponding
     * to an instance of <EMWEB_STRING>.
     */
    EmWebString_f       *emweb_string;

    /*
     * Archive-specific <EMWEB_INCLUDE> dispatch function.  Takes context
     * and index arguments.  Index selects the C-code fragment corresponding
     * to an instance of <EMWEB_INCLUDE>.
     */
    EmWebInclude_f      *emweb_include;

    /*
     * Archive-specific form table. The Form node index is hierarchical and
     * contains a form number and an element number.  The form table is
     * indexed by form number (0..n-1).  Each form table entry contains a
     * pointer to an array of field elements indexed by one less than the
     * element number.  (An element number of zero starts the form, and
     * an element number greater than the number of elements in the form
     * ends the form).  The table contains all the necessary information to
     * serve and submit forms.  The emweb_form_enum_table is a list of strings
     * corresponding to RADIO and single SELECT values.
     */
    const EwFormEntry *emweb_form_table;
    uint32      emweb_form_table_size;
    const char  **emweb_form_enum_table;
    uint32      emweb_form_enum_table_size;

    /*
     * Archive-specific imagemap table.  The table represents imagemaps
     * compiled by the EmWeb/Compiler into the archive.  This table is
     * indexed by a document node corresponding to the mapfile.  The
     * table entry contains a table of rectangular regions which are scanned
     * for matching x,y coordinates to find a match.
     */
    const EwImageMapTable *emweb_image_table;
    uint32              emweb_image_table_size;

    /*
     * Archive-specific CGI action dispatch table.
     */
```

```
ew_db.h        Sat Jun 28 07:59:50 1997        3
    const EmWebCGITable *emweb_cgi_table;
    uint32              emweb_cgi_table_size;

    /*
     * Reference count - this is initialized to zero by the EmWeb/Compiler,
     * but is writable (.data section).  This value is incremented for each
     * cloned page It is also incremented for each page currently in use by an
     * uncompleted request.  This prevents the application from attempting to
     * remove an archive that is in use.
     */
    uint32              reference_count;

    /*
     * Document list - this is the anchor of a list of loaded documents and
     * is used exclusively by EmWeb/Server to keep track of per-document nodes
     * created when loading the archive into the open hashed file system.
     * This should be set to EWS_DOCUMENT_NULL by the EmWeb/Compiler.
     */
    EwsDocument         document_list;

    /*
     * Archive-specific cookie table. This table is indexed by data_offset
     * value find in the cookie header node.
     *
     * !!!NOTE: location of this variable is important. For bacward comatibility
     *          of archives, it MUST be the last field in this structure.
     */
    const EwCookieAttributes *emweb_cookie_tbl;

} EwsArchive_t;

/*
 * DOCUMENT ARCHIVE DATA COMPONENT
 *
 * The document archive data component is a relocatable, endian-independent,
 * static data structure containing compressed documents and information about
 * how to access them.
 *
 * All 16-bit and 32-bit quantities are represented as arrays of octets.  The
 * first octet is the most significant (i.e. Big Endian).
 *
 * All pointers are 32-bit (array of four octets) offsets from the beginning of
 * the archive data.
 *
 * All strings are .asciiz ('\0'-terminated ASCII) and are represented by a
 * pointer to the string.
 *
 * Data storage for strings and various other variable-length components can
 * be placed between headers at the discretion of the EmWeb/Compiler.
 *
 * The document archive data component takes the following form:
 *
 *   EwsArchiveHeader
 *   (optional padding)
 *     [ EwsDocumentHeader [ EwsDocumentNode, ... ] (optional padding) ], ...
 *
 * The optional padding can be used to store variable length fields such
 * as strings, compressed documents, etc.
 */

/*
 * EwsArchiveHeader
 *
 * The archive data component begins with an archive header at offset zero
```

```
ew_db.h        Sat Jun 28 07:59:50 1997        4
 * NOTE WELL: KEEP SIZEOF_EWS_ARCHIVE_HEADER UP TO DATE!!!
 */
typedef struct EwsArchiveHeader_s
  {

    uint8       magic[4];       /* Magic number for archive validation */

#   define EW_ARCHIVE_MAGIC_0   'E'
#   define EW_ARCHIVE_MAGIC_1   'W'
#   define EW_ARCHIVE_MAGIC_2   'd'
#   define EW_ARCHIVE_MAGIC_3   'a'
#   define EW_ARCHIVE_MAGIC     (('E'<<24)|('W'<<16)|('d'<<8)|'a')

    uint8       version_maj;    /* Archive version (major/minor) */
    uint8       version_min;

    uint8       compiler_maj;   /* EmWeb/Compiler version (major/minor) */
    uint8       compiler_min;

    uint8       length[4];      /* length of archive, octets */
    uint8       name_offset[4]; /* Offset to archive name string */
    uint8       date_1123_offset[4]; /* Offset to HTTP date (RFC1123 format) */
    uint8       date_1036_offset[4]; /* Offset to HTTP date (RFC1036 format) */
    uint8       doc_offset[4];  /* Offset to first document header */
    uint8       dict_offset[4]; /* Offset to compression dictionary */

  } EwsArchiveHeader, * EwsArchiveHeaderP;

#define SIZEOF_EWS_ARCHIVE_HEADER 32  /* NEVER use sizeof() */

/*
 * EwsDocumentHeader
 *
 * For each document in the archive, there is a corresponding document
 * header.  The first document header is referenced by the doc_offset
 * field of the archive header.  Subsequent documents are chained together
 * using the next_offset field in each document header.  The last document
 * header in the archive contains a zero in the next_offset field.
 *
 * The following types of documents are supported:
 *
 *   Link       - Generates a redirection URL.  The redirection URL is an
 *                uncompressed string stored in the data area.  node_count
 *                should be zero.
 *
 *   Mime       - Generic (non-text) mime type.  The data area contains the mime
 *                document.  The mime content is indicated by the string at
 *                mime_offset.  Compression is optional.  node_count should be
 *                zero.
 *
 *   Text       - Generic text type.  The text may contain EmWeb/Compiler
 *                HTML extensions for <EMWEB_STRING>, <EMWEB_INCLUDE>,
 *                form get processing, etc.  Each such extension results in
 *                one or more document nodes being present describing the
 *                actions to take at certain points during the processing of
 *                the data.  The data contains the static parts of the
 *                document and may optionally be compressed.  The mime content
 *                is indicated by the string at mime_offset.
 *
 *   Index      - An index URL.  An absolute path URL is stored as an
 *                uncompressed string in the data area.  node_count should be
 *                zero.
 *
 *   Form       - A form action URL.  There should not be a data section.
 *                There will be a FORM node that indicates the index to
```

```
ew_db.h        Sat Jun 28 07:59:50 1997        5
 *                use for the start/set/submit operations in handling a
 *                posted form.
 *
 *    CGI         - A CGI URL.  There will be a CGI node that indicates the
 *                  index to use for the start/data operations in handling a
 *                  raw CGI request.  An optional data section containing
 *                  the uncompressed content of the corresponding URL file
 *                  (if present) is made available in the EwsContext.  This
 *                  could be used for CGI imagemap data, for example.
 *
 *    Imagemap    - A mapfile URL.  There will be an imagemap node that
 *                  indicates the index into a per-archive imagemap table
 *                  for processing an imagemap.
 *
 * Note that any document might be marked hidden.  If so, the document is
 * not accessible by a client browser directly.  Instead, the document is
 * used as a building block.  It can be cloned, or EMWEB_INCLUDEd.
 *
 * NOTE WELL: KEEP SIZEOF_EWS_DOCUMENT_HEADER UP TO DATE!!
 */
typedef struct EwsDocumentHeader_s
  {
    uint8       next_offset[4]; /* offset to next document or 0 if last */
    uint8       url_offset[4];  /* offset to URL string */

    uint8       document_type;  /* type of document */

    /* If you add a document type, update ../compiler/readarchive.c */
#   define EW_ARCHIVE_DOC_TYPE_MASK      0x0F    /* type field */
#   define EW_ARCHIVE_DOC_TYPE_LINK      0       /* redirection URL */
#   define EW_ARCHIVE_DOC_TYPE_MIME      1       /* generic mime data type */
    /*     EW_ARCHIVE_DOC_TYPE_TEXT      2        * [obsolete] */
#   define EW_ARCHIVE_DOC_TYPE_FORM      4       /* Form action URL */
#   define EW_ARCHIVE_DOC_TYPE_INDEX     5          /* index URL */
#   define EW_ARCHIVE_DOC_TYPE_CGI       8       /* CGI-BIN interface */
#   define EW_ARCHIVE_DOC_TYPE_IMAGEMAP  9       /* Imagemap interface */
#   define EW_ARCHIVE_DOC_TYPE_FILE      10      /* local filesystem */
#   define EW_ARCHIVE_DOC_FLAG_STATIC    0x20    /* treat as static document */
#   define EW_ARCHIVE_DOC_FLAG_FORM      0x40    /* contains form node */
#   define EW_ARCHIVE_DOC_FLAG_HIDDEN    0x80    /* hidden (internal use) */

    uint8       comp_method;    /* compression method */

#   define EW_ARCHIVE_COMPRESSION_NONE  0
#   define EW_ARCHIVE_COMPRESSION_EMWEB 1
#   define EW_ARCHIVE_COMPRESSION_DEFAULT EW_ARCHIVE_COMPRESSION_EMWEB

    uint8       node_count[2];  /* number of document nodes */
    uint8       node_offset[4]; /* offset to array of document nodes */

    uint8       mime_offset[4]; /* offset to mime content type string */
    uint8       realm_offset[4];/* offset to authentication realm string */

    uint8       orig_length[4]; /* length of original (uncompressed) data */
    uint8       data_length[4]; /* length of compressed data */
    uint8       data_offset[4]; /* offset to compressed data */

    uint8       hdr_node_count; /* number of document header nodes */
    uint8       hdr_flags;      /* header specific flags              */

#   define EW_ARCHIVE_DOC_HDR_COOKIE_FLG   0x01    /* set if COOKIE hdr present */
#   define EW_ARCHIVE_DOC_HDR_CACHE_FLG    0x02    /* set if CACHE hdr present */
#   define EW_ARCHIVE_DOC_HDR_VARY_FLG     0x03    /* set if VARY hdr present */

    uint8       reserved[2];
```

```
ew_db.h        Sat Jun 28 07:59:50 1997        6
  } EwsDocumentHeader, * EwsDocumentHeaderP;

#define SIZEOF_EWS_DOCUMENT_HEADER 40    /* never use sizeof */

/*
 * ewsDocumentNode
 *
 * The document header points to an array of zero or more document nodes
 * followed by header nodes.
 * The node_count field of the document header indicates the number of nodes
 * present.  Each node represents a special run-time action to be taken by the
 * EmWeb/Server when processing requests for the document.  When processing
 * reaches the specified point in the uncompressed data, the specified action
 * is taken.
 *
 * Header node contains information needed to create HTTP reply header(s).
 * The hdr_count field of the document header indicates the number of hdr_nodes
 * present.  Each node represents a special run-time action to be taken by the
 * EmWeb/Server when processing requests for the document. Header nodes are processed
 * first as the headers are built into the reply befor data part of the document
 * is processed.
 * Header nodes are sorted in the archive by type. Compiler sorts and includes
 * in the archive only headers of the types for which the flag is set in hdr_flags
 * field in the Document Header.
 *
 * The document header node(s) immediately follow the ewsDocumentNodes for a
 * given document.
 * For a cache control header, the first offset in the node will be used to
 * point to the ETAG for this (static) document. ETAGS are used for
 * cache-control in HTTP 1.1. (Note that an ETAG will not be used for a
 * dynamic document).
 * The type field will indicate what type of header node it is - cache or
 * state cookie, etc.
 * The other fields are reserved.
 * Note that in the future, we may use the last 32 bits in this header
 * to be an instance C pointer for the cache control header.

 * NOTE WELL:  KEEP SIZEOF_EWS_DOCUMENT_NODE UP TO DATE!!!
 */
typedef struct EwsDocumentNode_s
  {
    uint8       data_offset[4]; /* location of node in uncompressed data
                                   stream (relative to start of uncompressed
                                   document data),
                                   for COOKIE header it is an index to CookieTable
                                */

    uint8       type;           /* type of node */

    /* If you add a node type, update ../compiler/readarchive.c */
#   define EW_DOCUMENT_NODE_TYPE_STRING         1       /* <EMWEB_STRING> */
#   define EW_DOCUMENT_NODE_TYPE_INCLUDE        2       /* <EMWEB_INCLUDE> */
#   define EW_DOCUMENT_NODE_TYPE_FORM           3       /* <FORM> operation */
#   define EW_DOCUMENT_NODE_TYPE_CGI            4       /* CGI operation */
#   define EW_DOCUMENT_NODE_TYPE_IMAGEMAP       5       /* Image Map */
#   define EW_DOCUMENT_NODE_TYPE_COOKIE         6       /* <EMWEB_COOKIE> */
#   define EW_DOCUMENT_NODE_TYPE_CACHE          7       /* CACHE */
#   define EW_DOCUMENT_NODE_TYPE_VARY           8       /* content negotiation*/

    uint8       attributes;     /* node attributes */

    /* If you add a node attribute, update ../compiler/readarchive.c */
#   define EW_DOCUMENT_NODE_REPEAT      0x01    /* repeat until NULL for
                                                   EMWEB_STRING and
                                                   EMWEB_INCLUDE nodes */
```

```
ew_db.h        Sat Jun 28 07:59:50 1997        7
#   define EW_DOCUMENT_HDR_SECURE        0x02    /* SECURE flag for cookie */

    uint8       reserved[2];    /* reserved for future use */

    uint8       index[4];       /* index assigned to node.  This may be
                                   hierarchical.  (e.g. the FORM index
                                   contains a form number and element
                                   number.  the STRING index contains
                                   a conversion type (most significant
                                   8 bits) and a string number (least
                                   significant 24 bits).
                                   The COOKIE index contains string number
                                   for cookie's value generating C-code.
                                   This index is applied to the switch contained
                                   in emweb_string() function */

  } EwsDocumentNode, * EwsDocumentNodeP;

#define SIZEOF_EWS_DOCUMENT_NODE  12    /* never use sizeof */
#define NEXT_DOCNODE( x ) ((EwsDocumentNodeP)(((uint8 *)(x)) \
                                          + SIZEOF_EWS_DOCUMENT_NODE))
#endif /* _EW_DB_H */
```

```
ew_form.h       Sat Jun 28 07:59:50 1997      1
/*
 * $Id: ew_form.h,v 1.23 1997/06/28 11:59:50 ian Exp $
 *
 * Product: EmWeb
 * Release: $Name:  $
 *
 * %CopyRight%
 *
 * EmWeb Form definitions between server and generated code.  These are
 * also used in typed EMWEB_STRINGs, etc.
 *
 */
#ifndef _EW_FORM_H
#define _EW_FORM_H

#include "ew_types.h"
#include "ew_common.h"
#include "ews_def.h"

/*
 * Application Form Serve Function
 * Fills in form template with default values.  (We comment out the prototype
 * to avoid compiler warnings -- the actual application functions define
 * a specific form structure instead of void*).
 *
 * context - request context
 * form    - pointer to form-specific structure
 *
 * No return value
 */
#ifndef __cplusplus
typedef void EwaFormServe_f ( );
#else /* __cplusplus */
typedef void EwaFormServe_f ( EwsContext context, void * form );
#endif /* __cplusplus */

/*
 * Application Form Submit Function
 * Processes submitted form.  (We comment out the prototype
 * to avoid compiler warnings -- the actual application functions define
 * a specific form structure instead of void*).
 *
 * context - request context
 * form    - pointer to form-specific structure
 *
 * Returns NULL for default response to server (accepted), or redirection URL.
 */
#ifndef __cplusplus
typedef char * EwaFormSubmit_f ( );
#else /* __cplusplus */
typedef char * EwaFormSubmit_f ( EwsContext context, void * form );
#endif /* __cplusplus */

/*
 * Form Field Types
 * Each field element has a type associated with it.  Each type has conversion
 * routines to handle conversions between HTML and internal representations.
 * NOTE: if you modify this enum list, you MUST update the FieldMap in
 * file ewc_code.c!!
 */
typedef enum EwFieldType_e
  {
    ewFieldTypeDecimalOverride = 0 /* unsigned decimal integer */
   ,ewFieldTypeRadio             /* radio button - serve setup and submit */
   ,ewFieldTypeRadioField        /* radio fields - used in serving form */
```

ew_form.h Sat Jun 28 07:59:50 1997 2

```
    ,ewFieldTypeSelect          /* select enum - serve setup and submit */
    ,ewFieldTypeSelectField     /* select enum fields - used in serving form */
    ,ewFieldTypeSelectMulti     /* multiple select */
    ,ewFieldTypeCheckbox        /* checkbox */
    ,ewFieldTypeText            /* text field */
    ,ewFieldTypeImage           /* image field */
    ,ewFieldTypeDecimalUint     /* unsigned integer */
    ,ewFieldTypeDecimalInt      /* signed integer */
    ,ewFieldTypeHexInt          /* hexadecimal integer */
    ,ewFieldTypeHexString       /* hexadecimal string */
    ,ewFieldTypeDottedIP        /* dotted IP address */
    ,ewFieldTypeIEEEMAC         /* IEEE MAC */
    ,ewFieldTypeFDDIMAC         /* FDDI MAC */
    ,ewFieldTypeStdMAC          /* Standard MAC */
    ,ewFieldTypeDecnetIV        /* Decnet IV Address */
    ,ewFieldTypeFile            /* input type file handle */
    ,ewFieldTypeTextOverride    /* serve input file name field */
    ,ewFieldTypeObjectID        /* object identifier */
    ,ewFieldTypeMAX             /* MUST BE LAST */
  } EwFieldType;

/*
 * Form Field Representation
 * The compiler generates the following structure for each element in the
 * form.  In general, a FORM document node is placed in the archive at the
 * offset immediately following a VALUE= in TEXT, TEXTAREA, HIDDEN, or PASSWORD
 * field.  The field component of the form node index refers to a field
 * structure as defined below.  The field type would be ewFieldTypeText, or
 * one of the derivative extended types such as ewFieldToDottedIP.  If a
 * VALUE was specified in the source HTML, the compiler translates it into
 * the appropriate internal representation and creates the default_value of
 * the field.  (For ewFieldTypeText, this is simply a pointer to the null
 * terminated string.  For other fields, this is a pointer to a value whose
 * length is determined by the type in ewFieldTable).  The NAME= part of
 * the field is saved, and the offset into the compiler-generated form for
 * the value and status portions of the field is stored in value_offset and
 * status_offset.
 *
 * Checkboxes and multiple selections are fairly straight forward.
 * The type ewFieldCheckbox or ewFieldSelectMulti is used, and the
 * node appears where the word "CHECKED" or "SELECTED" may appear.  If a
 * VALUE= is present, it is saved in the sel_value field.  Otherwise, these
 * types are identical to the extended text types above.
 *
 * Radio buttons and single-select boxes are more complex.  With a single
 * NAME=, there may be multiple VALUE= where only one applies to the actual
 * value of the form field.  With N radio buttons or single select options,
 * the compiler generates N+1 field structures.  One structure is used to
 * read and write the value in the internal generated form.  This would have
 * type ewFieldTypeRadio or ewFieldTypeSelect, contain the default value,
 * offsets into the form structure, and the field name.  In addition,an array of
 * integers containing all values corresponding to the generated enumerator is
 * saved in enum_list.  The other N field structures would represent each radio
 * button or select option and have the type ewFieldTypeRadioField or
 * ewFieldTypeSelectField.  These are referenced by form nodes in the document
 * archive at the location corresponding to where a "CHECKED" or "SELECTED"
 * keyword would be inserted.  They would not have a name field.  Instead,
 * the enum_value field would be set to the VALUE= enumeration corresponding
 * to the specific field.
 */
typedef struct EwFormField_s
  {
    EwFieldType         field_type;     /* type of field */
    const void          *default_value; /* pointer to default value */
    uintf               value_offset;   /* offset to value in form struct */
```

ew_form.h Sat Jun 28 07:59:50 1997 3

```
    uintf               status_offset;  /* offset to status in form struct */
    const char          *name;          /* name of field from NAME="" */
    const int           *enum_list;     /* list of field values by enum */
    int                 enum_value;     /* SELECT/RADIO value of field */
    const char          *sel_value;     /* SELECT multiple value field */
  ] EwFormField, * EwFormFieldP;

/*
 * Form Representation
 * The following structure represents an HTML FORM.  This points to the
 * application-provided serve and submit functions, indicates the size
 * of the compiler-generated form structure, and points to a table of
 * field structures as defined above.
 */
typedef struct EwFormEntry_s
  {
    EwaFormServe_f      *serve_f;       /* application serve function */
    EwaFormSubmit_f     *submit_f;      /* application submit function */
    uintf               form_size;      /* sizeof form structure */
    uintf               field_count;    /* number of fields in form */
    const EwFormField   *field_list;     /* list of fields */
  ] EwFormEntry, * EwFormEntryP;

/*
 * Document Node Conversion Macros
 * These macros convert between the 32-bit node index and a form and element
 * index.
 */
#define EW_FORM_ELEMENT_START                   (0)
#define EW_FORM_ELEMENT_END                     (0xffff)
#define EW_FORM_NODE_INDEX_TO_FORM_INDEX(i)     ((i) >> 16)
#define EW_FORM_NODE_INDEX_TO_ELEMENT_INDEX(i)  ((i) & 0xffff)
#define EW_FORM_ELEMENT_TO_NODE_INDEX(f, e)     (((f) << 16) | (e))

/*
 * Form conversion functions.
 * The "To" function converts to HTML ASCII format from the internal
 * representation and writes it directly to the output data stream.
 * The "From" function converts from HTML ASCII contained in a parsed network
 * buffer EwsString and converts it to the internal representation.
 */
typedef void EwFieldTypeTo_f
  ( EwsContext context, const EwFormField *field );
typedef void EwFieldTypeFrom_f
  ( EwsContext context, const EwFormField *field, EwsStringP stringp );
typedef void EwFieldTypeFree_f ( void *p );

#ifdef EW_CONFIG_OPTION_FIELDTYPE_RADIO
extern EwFieldTypeTo_f          ewFieldToRadioField;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_RADIO */

#if defined(EW_CONFIG_OPTION_FIELDTYPE_RADIO)\
 || defined(EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE)
extern EwFieldTypeFrom_f        ewFieldFromRadio;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_RADIO | SELECT_SINGLE */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE
extern EwFieldTypeTo_f          ewFieldToSelectField;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_SELECT_MULTIPLE
extern EwFieldTypeTo_f          ewFieldToSelectMulti;
extern EwFieldTypeFrom_f        ewFieldFromSelectMulti;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_SELECT_MULTIPLE */
```

ew_form.h Sat Jun 28 07:59:50 1997 4

```
#ifdef EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX
extern EwFieldTypeTo_f          ewFieldToCheckbox;
extern EwFieldTypeFrom_f        ewFieldFromCheckbox;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX
extern EwFieldTypeTo_f          ewFieldToText;
extern EwFieldTypeFrom_f        ewFieldFromText;
extern EwFieldTypeFree_f        ewFieldFreeText;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX */

extern EwFieldTypeTo_f          ewFieldToDecimalUint;

#ifdef EW_CONFIG_OPTION_FIELDTYPE_IMAGE
extern EwFieldTypeFrom_f        ewFieldFromImage;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_IMAGE */

#if defined(EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_UINT)\
 || defined(EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_INT)\
 || defined(EW_CONFIG_OPTION_FIELDTYPE_IMAGE)
extern EwFieldTypeFrom_f        ewFieldFromDecimal;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_UINT|INT|IMAGE */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_INT
extern EwFieldTypeTo_f          ewFieldToDecimalInt;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_INT */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_HEX_INT
extern EwFieldTypeTo_f          ewFieldToHexInt;
extern EwFieldTypeFrom_f        ewFieldFromHexInt;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_HEX_INT */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING
extern EwFieldTypeTo_f          ewFieldToHexString;
extern EwFieldTypeFrom_f        ewFieldFromHexString;
extern EwFieldTypeFree_f        ewFieldFreeHexString;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_DOTTEDIP
extern EwFieldTypeTo_f          ewFieldToDottedIP;
extern EwFieldTypeFrom_f        ewFieldFromDottedIP;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_DOTTEDIP */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_DECNET_IV
extern EwFieldTypeTo_f          ewFieldToDecnetIV;
extern EwFieldTypeFrom_f        ewFieldFromDecnetIV;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_DECNET_IV */

#if defined(EW_CONFIG_OPTION_FIELDTYPE_IEEE_MAC)\
 || defined(EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC)\
 || defined(EW_CONFIG_OPTION_FIELDTYPE_STD_MAC)
extern EwFieldTypeFrom_f        ewFieldFromMAC;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_IEEE|FDDI|STD_MAC */

#if defined(EW_CONFIG_OPTION_FIELDTYPE_IEEE_MAC)\
 || defined(EW_CONFIG_OPTION_FIELDTYPE_STD_MAC)
extern EwFieldTypeTo_f          ewFieldToIEEEMAC;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_IEEE|STD_MAC */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC
extern EwFieldTypeTo_f          ewFieldToFDDIMAC;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_STD_MAC
extern EwFieldTypeFrom_f        ewFieldFromStdMAC;
```

ew_form.h Sat Jun 28 07:59:50 1997 5

```
#endif /* EW_CONFIG_OPTION_FIELDTYPE_STD_MAC */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_FILE
extern EwFieldTypeFrom_f        ewFieldFromFile;
extern EwFieldTypeFree_f        ewFieldFreeFile;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_FILE */

#ifdef EW_CONFIG_OPTION_FIELDTYPE_OID
extern EwFieldTypeTo_f          ewFieldToObjectID;
extern EwFieldTypeFrom_f        ewFieldFromObjectID;
extern EwFieldTypeFree_f        ewFieldFreeObjectID;
#endif /* EW_CONFIG_OPTION_FIELDTYPE_OID */

typedef struct EwFormFieldTable_s
  {
    uintf                       field_size;     /* size of field */
    EwFieldTypeTo_f             *to_f;          /* convert to HTML */
    EwFieldTypeFrom_f           *from_f;        /* convert from HTML */
    EwFieldTypeFree_f           *free_f;        /* free dynamic pointer */
  } EwFormFieldTable;

#ifdef EW_DEFINE_FIELD_TYPE_TABLE
const EwFormFieldTable ewFormFieldTable[] =
  {

    { sizeof(uint32),  ewFieldToDecimalUint,    NULL,                   NULL }

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_RADIO
   ,{ sizeof(int),     NULL,                    ewFieldFromRadio,       NULL }
   ,{ 0,               ewFieldToRadioField,     NULL,                   NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_RADIO */
   ,{ 0,               NULL,                    NULL,                   NULL }
   ,{ 0,               NULL,                    NULL,                   NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_RADIO */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE
   ,{ sizeof(int),     NULL,                    ewFieldFromRadio,       NULL }
   ,{ 0,               ewFieldToSelectField,    NULL,                   NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE */
   ,{ 0,               NULL,                    NULL,                   NULL }
   ,{ 0,               NULL,                    NULL,                   NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_SELECT_SINGLE */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_SELECT_MULTIPLE
   ,{ sizeof(boolean), ewFieldToSelectMulti, ewFieldFromSelectMulti,NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_SELECT_MULTIPLE */
   ,{ 0,               NULL,                    NULL,                   NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_SELECT_MULTIPLE */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX
   ,{ sizeof(boolean), ewFieldToCheckbox,    ewFieldFromCheckbox,    NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX */
   ,{ 0,               NULL,                    NULL,                   NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_CHECKBOX */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_TEXT
   ,{ sizeof(char *),  ewFieldToText,          ewFieldFromText,ewFieldFreeText }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_TEXT */
   ,{ 0,               NULL,                    NULL,                   NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_TEXT */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_IMAGE
   ,{ sizeof(uint32),  NULL,           ewFieldFromDecimal,  NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_IMAGE */
   ,{ 0,               NULL,                    NULL,                   NULL }
```

ew_form.h Sat Jun 28 07:59:50 1997 6

```
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_IMAGE */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_UINT
   ,{ sizeof(uint32),   ewFieldToDecimalUint,   ewFieldFromDecimal,  NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_UINT */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_UINT */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_INT
   ,{ sizeof(uint32),   ewFieldToDecimalInt,    ewFieldFromDecimal,  NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_INT */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_DECIMAL_INT */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_HEX_INT
   ,{ sizeof(uint32),   ewFieldToHexInt,        ewFieldFromHexInt,   NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_HEX_INT */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_HEX_INT */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING
   ,{ sizeof(EwsFormFieldHexString), ewFieldToHexString, ewFieldFromHexString,
      ewFieldFreeHexString }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_HEX_STRING */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_DOTTEDIP
   ,{ sizeof(uint32),   ewFieldToDottedIP,      ewFieldFromDottedIP, NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_DOTTEDIP */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_DOTTEDIP */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_IEEE_MAC
   ,{ 6,                ewFieldToIEEEMAC,       ewFieldFromMAC,      NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_IEEE_MAC */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_IEEE_MAC */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC
   ,{ 6,                ewFieldToFDDIMAC,       ewFieldFromMAC,      NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_FDDI_MAC */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_STD_MAC
   ,{ 6,                ewFieldToIEEEMAC,       ewFieldFromStdMAC,   NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_STD_MAC */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_STD_MAC */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_DECNET_IV
   ,{ sizeof(uint16),   ewFieldToDecnetIV,      ewFieldFromDecnetIV, NULL }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_DECNET_IV */
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_DECNET_IV */

#  ifdef EW_CONFIG_OPTION_FIELDTYPE_FILE
   ,{ sizeof(EwaFileHandle), NULL,          ewFieldFromFile,ewFieldFreeFile }
   ,{ sizeof(char *),   ewFieldToText,      ewFieldFromText,ewFieldFreeText }
#  else  /* !EW_CONFIG_OPTION_FIELDTYPE_FILE */
   ,{ 0,                NULL,                   NULL,                NULL }
   ,{ 0,                NULL,                   NULL,                NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_FILE */
```

```
ew_form.h       Sat Jun 28 07:59:50 1997       7
#  ifdef EW_CONFIG_OPTION_FIELDTYPE_OID
   ,{ sizeof(EwsFormFieldObjectID), ewFieldToObjectID, ewFieldFromObjectID,
      ewFieldFreeObjectID }
#  else /* EW_CONFIG_OPTION_FIELDTYPE_OID */
   ,{ 0,                  NULL,                  NULL,                  NULL }
#  endif /* EW_CONFIG_OPTION_FIELDTYPE_OID */

  };
#else /* EW_DEFINE_FORM_FIELD_TYPE_TABLE */
extern const EwFormFieldTable ewFormFieldTable[];
#endif /* EW_DEFINE_FORM_FIELD_TYPE_TABLE */

#endif /* _EW_FORM_H */
```

ew_imap.h Sat Jun 28 07:58:56 1997 1

```
/*
 * $Id: ew_imap.h,v 1.7 1997/05/21 21:55:48 ian Exp $
 *
 * Product: EmWeb
 * Release: $Name:  $
 *
 * %CopyRight%
 *
 * EmWeb image map definitions between server and generated code
 *
 */
#ifndef _EW_IMAP_H
#define _EW_IMAP_H

#include "ew_types.h"

typedef struct EwImageMap_s
  {
    uint32      ul_x;   /* upper-left X coordinate */
    uint32      ul_y;   /* upper-left Y coordinate */
    uint32      lr_x;   /* lower-right X coordinate */
    uint32      lr_y;   /* lower-right Y coordinate */
    const char  *url;   /* relative redirection URL */
  } EwImageMap, * EwImageMapP;

typedef struct EwImageMapTable_s
  {
    const EwImageMap *map_table; /* table of map entries */
    uintf       map_entries;    /* size of map_table */
    const char  *default_url;   /* default URL, or NULL if not specified */
  } EwImageMapTable, * EwImageMapTableP;

#endif /* _EW_IMAP_H */
```

CLAIMS

What is claimed is:

1. A method for providing a graphical user interface having dynamic elements, comprising the steps of:

defining elements of the graphical user interface in at least one text document written in a mark-up language;

including at a location in the document a code tag containing a segment of application source code;

serving the text document to a client which interprets the mark-up language; and

when the location is encountered, serving to the client a sequence of characters derived from a result of executing a sequence of instructions represented by the segment of application source code.

2. The method of claim 1, further comprising the step of:

including in the document at least one more code tag containing a segment of application source code.

3. The method of claim 1, wherein the step of defining further comprises the step of:

providing a plurality of documents which collectively define the graphical user interface; and

storing the text document and the plurality of documents as files in a directory tree.

4. The method of claim 3, further comprising the steps of:

compiling the directory tree and the files therein into an archive including content sources; and

decompiling a content source back into the text document before the step of serving.

5. A method for providing a graphical user interface having dynamic elements, comprising the steps of:

defining elements of the graphical user interface in at least one text document written in a mark-up language;

including in the document a string identified by prototype tags;

serving the text document to a prototyping client which interprets the mark-up language but does not recognize and does not display the prototype tags, but does display the string; and

compiling the text document into a content source, omitting the prototype tags and the string identified thereby.

6. The method of claim 5, further comprising the step of:

including at a location in the document a code tag containing a segment of application source code, wherein the prototyping client does not recognize and does not display the code tag; and

the step of further comprising including the segment of application source code in the content source.

7. The method of claim 6, further comprising the steps of:

decompiling the content source into a replica of the text document;

serving the replica to a user client; and

when the location is encountered in the replica, serving to the user client a character stream derived from a result generated by executing the segment of application source code.

8. The method of claim 7, further comprising the steps of:

including in the document at least one more code tag containing a segment of application source code; and

including in the document at least one more string identified by code tags.

9. A method for providing a graphical user interface having dynamic elements, comprising the steps of:

defining elements of the graphical user interface in at least one text document written in a mark-up language;

including at a location in the document a code tag containing a segment of application source code;

including in the document a string identified by prototype tags;

compiling the text document into a content source;

decompiling the content source into a replica of the text document;

serving the replica of the text document to a client which interprets the mark-up language; and

when the location is encountered in the replica, serving to the client a character stream generated by executing the segment of application source code.

10. The method of claim 9, further comprising the steps of:

including in the document at least one more code tag containing a segment of application source code; and

including in the document at least one more string identified by code tags.

11. A software product recorded on a medium, the software product comprising a mark-up language compiler which can compile a mark-up language document into a data structure in a native application programming language, the compiler recognizing one or more code tags which designate included text as a segment of application source code to be saved in a file for compilation by a compiler of the native application programming language.

12. A method for providing a graphical user interface having displayed forms for entry of data, comprising the steps of:

defining elements of the graphical user interface in at least one text document written in a mark-up language;

naming in the document a data item requested of a user and used by an application written in a native application programming language; and

compiling the text document into a content source including a data structure definition in the native application programming language for the named data item.

13. The method of claim 12, wherein the text document is viewed using a prototype client during development and is viewed using a user client during use, further comprising the steps of:

naming the data item within a tag understood in the mark-up language for viewing by both the prototype client and the user client; and

including in the content source a definition of a function to provide a default value for the data item when the content source is served and a definition of a function to perform an application specific operation when the data item is returned by the user client.

14. In a computer-based apparatus for developing a graphical user interface for an application, the apparatus including an editor which can manipulate a document written in a mark-up language and a viewer which can display a document written in the mark-up language. the apparatus further comprising:

a mark-up language compiler which recognizes a code tag containing a source code fragment in a native application source code language, the code tag not otherwise part of the mark-up language, the compiler producing as an output a representation in the native application source code language of the document, including a copy of the source code fragment.

15. The apparatus of claim 14, wherein the mark-up language compiler further recognizes begin and end prototype tags not otherwise part of the mark-up language, the output representation omitting the begin and end prototype tags and all content appearing therebetween in the document.

16. A method for developing and prototyping graphic user interfaces for an application comprising the steps of:

accessing an HTML file,

encapsulating portions of said HTML and entering source code therein,

producing a source module from said HTML with encapsulated portions,

producing source code for a server, and

cross compiling and linking said application, said source code module and said server thereby producing executable object code.

17. The method of claim 16, further comprising the steps of:

running said object code,

executing said compiled encapsulated code when requested by a viewer, wherein said encapsulated code is associated with said application.

18. The method of claim 17, further comprising the steps of:

converting data returned by execution of said compiled encapsulated code into a form displayable by said viewer.

19. The method of claim 18, wherein the data returned by executing said compiled encapsulated code changes over time as a result of changes within the application.

20. A data structure fixed in a computer readable medium, the data structure for use in a computer system including a client and a server in communication with each other, the data structure comprising:

cross-compiled, stored and linked, HTML files with encapsulated portions containing executable code associated with said application, server code, and application code, wherein said executable code is run when the HTML file is served thereby providing real time dynamic data associated with said application.

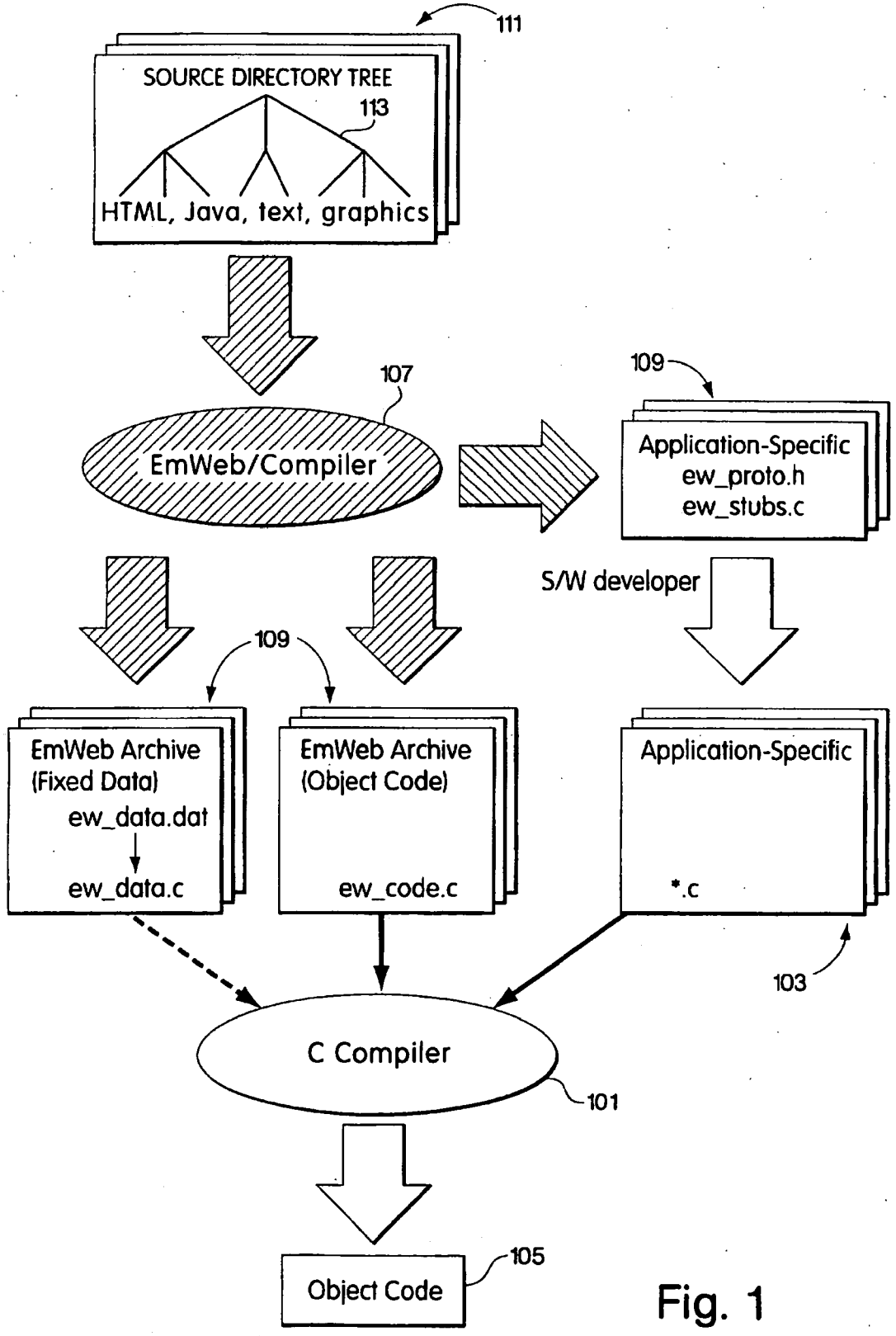

111
SOURCE DIRECTORY TREE
113
HTML, Java, text, graphics
107
EmWeb/Compiler
109
Application-Specific
ew_proto.h
ew_stubs.c
S/W developer
109
EmWeb Archive
(Fixed Data)
ew_data.dat
ew_data.c
EmWeb Archive
(Object Code)
ew_code.c
Application-Specific
*.c
103
C Compiler
101
Object Code
105

Fig. 1

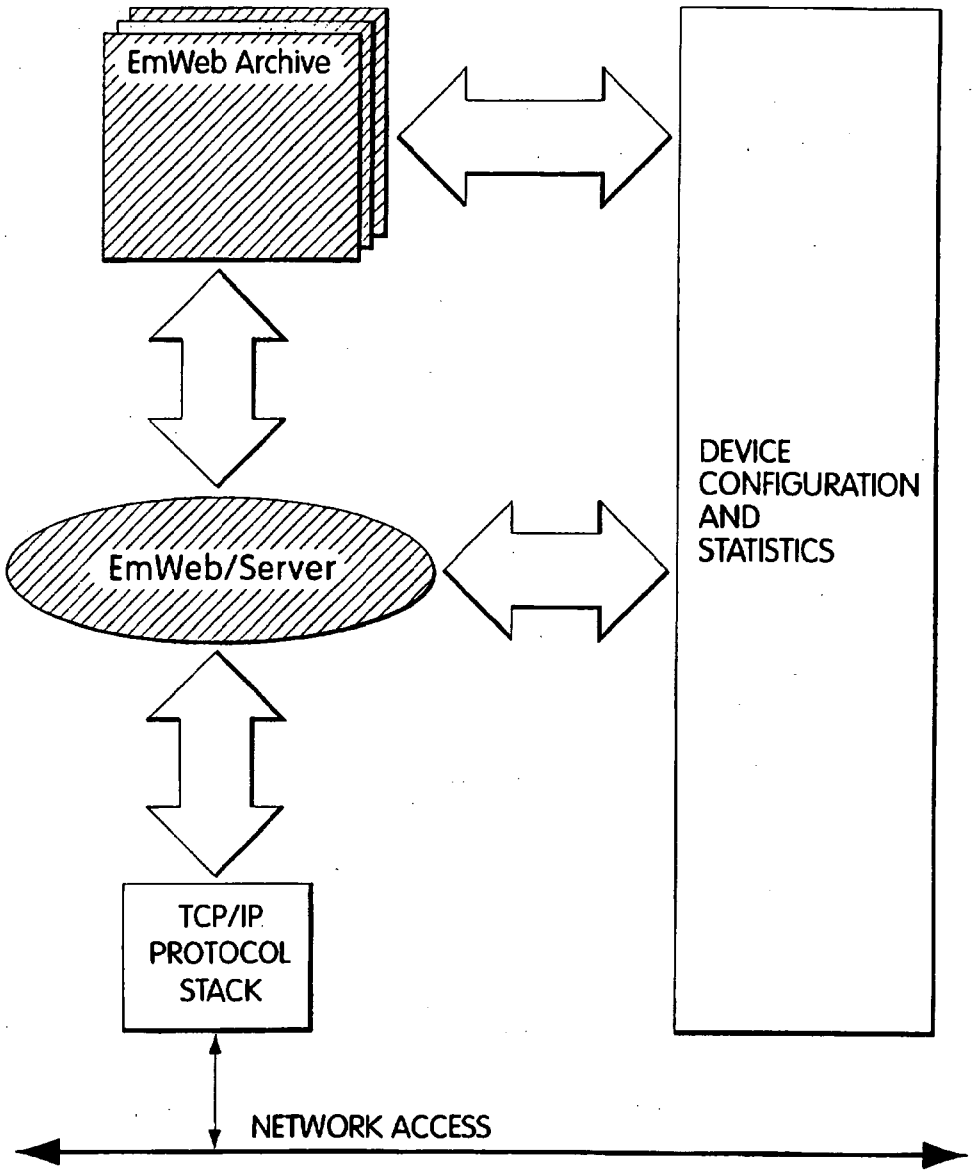

EmWeb Archive
EmWeb/Server
TCP/IP
PROTOCOL
STACK
DEVICE
CONFIGURATION
AND
STATISTICS
NETWORK ACCESS

Fig. 2

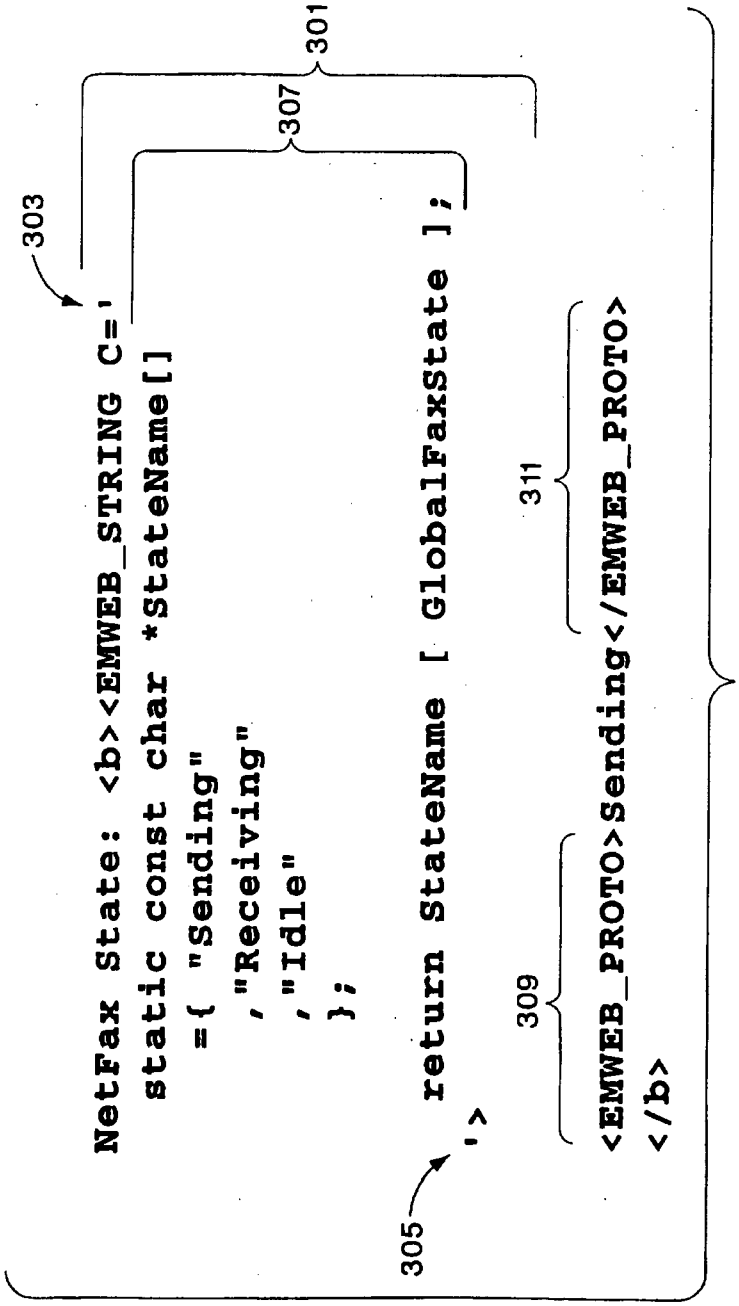

301
307
303
NetFax State: <EMWEB_STRING C='
static const char *StateName[]
={ "Sending"
,"Receiving"
,"Idle"
};
return StateName [GlobalFaxState];
'>
305
309
311
<EMWEB_PROTO>Sending</EMWEB_PROTO>

Fig. 3

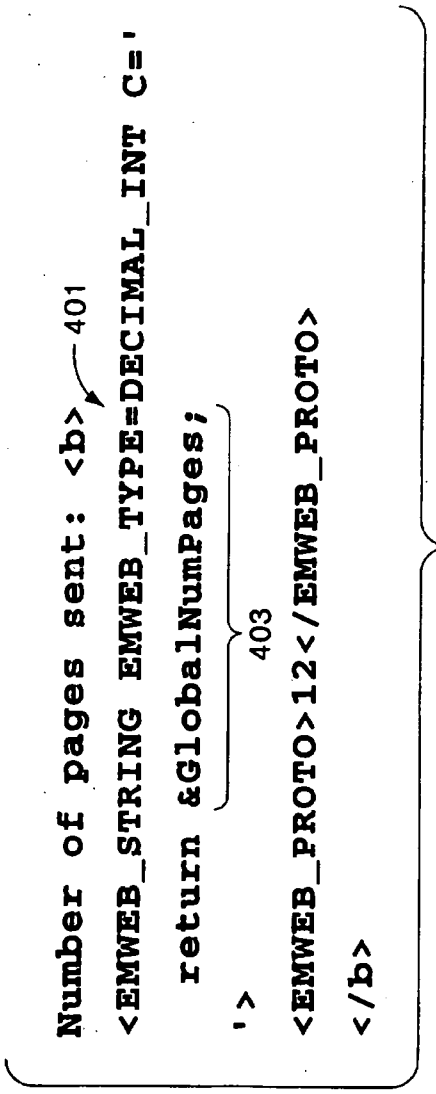

Number of pages sent:
401
<EMWEB_STRING EMWEB_TYPE=DECIMAL_INT C='
return &GlobalNumPages;
'>
403
<EMWEB_PROTO>12</EMWEB_PROTO>

Fig. 4

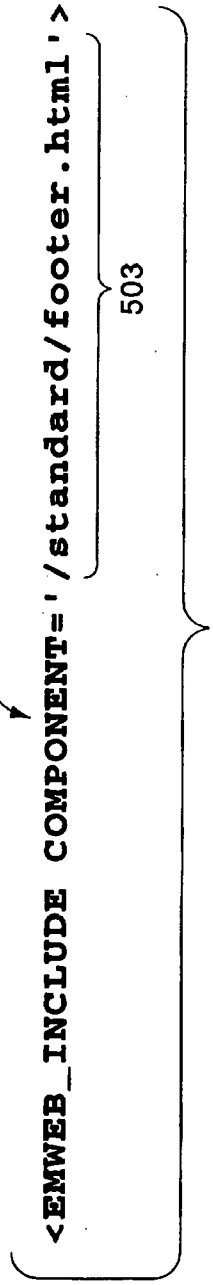


Fig. 5

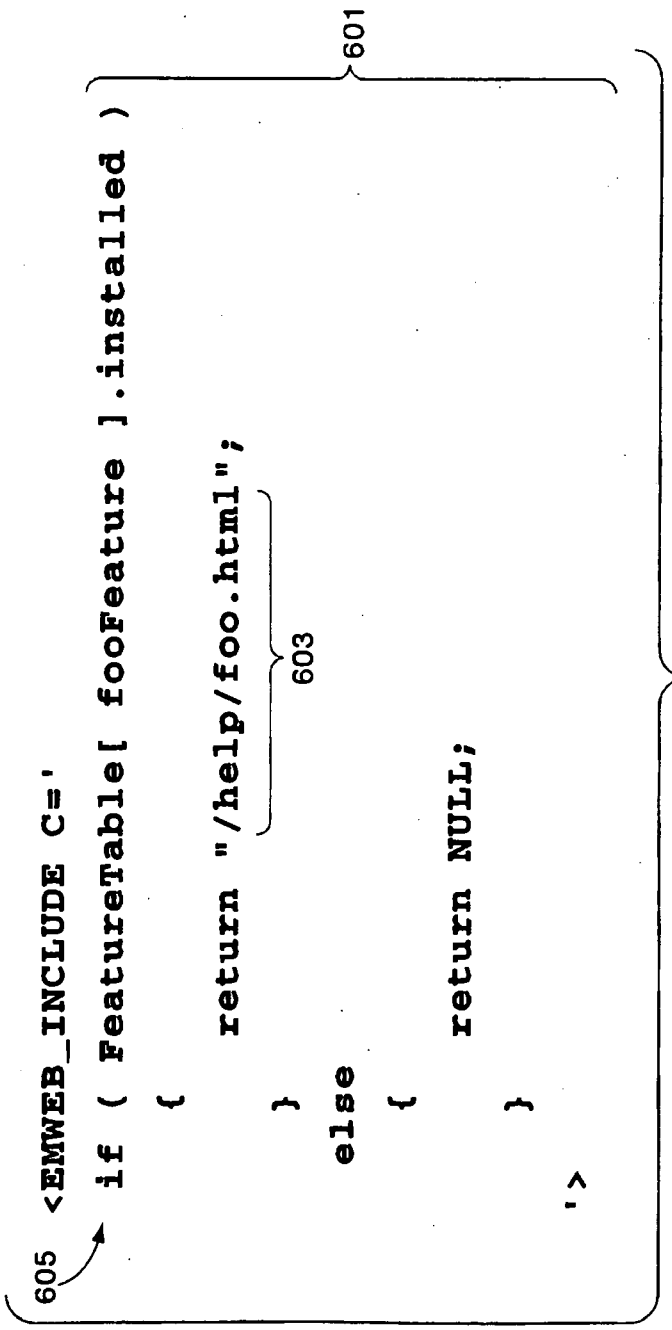
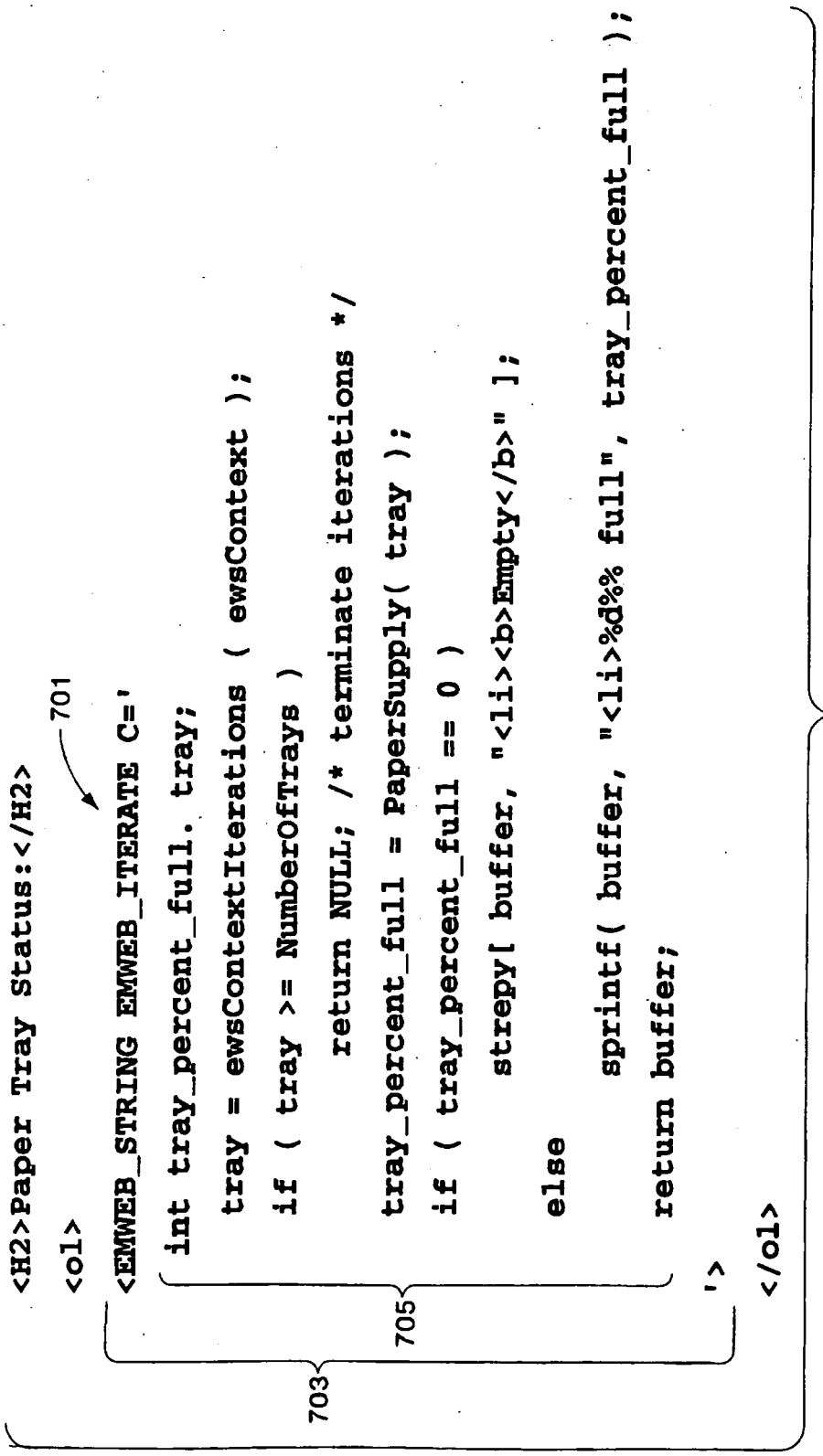

601
605
603
<EMWEB_INCLUDE C='
if (FeatureTable[fooFeature].installed)
{
return "/help/foo.html";
}
else
{
return NULL;
}
'>

Fig. 6


<H2>Paper Tray Status:</H2>

<EMWEB_STRING EMWEB_ITERATE C='
int tray_percent_full. tray;
tray = ewsContextIterations (ewsContext);
if (tray >= NumberOfTrays)
return NULL; /* terminate iterations */
tray_percent_full = PaperSupply(tray);
if (tray_percent_full == 0)
strepy[buffer, "<li>Empty"];
else
sprintf(buffer, "<li>%d%% full", tray_percent_full);
return buffer;
'>

701
703
705

Fig. 7

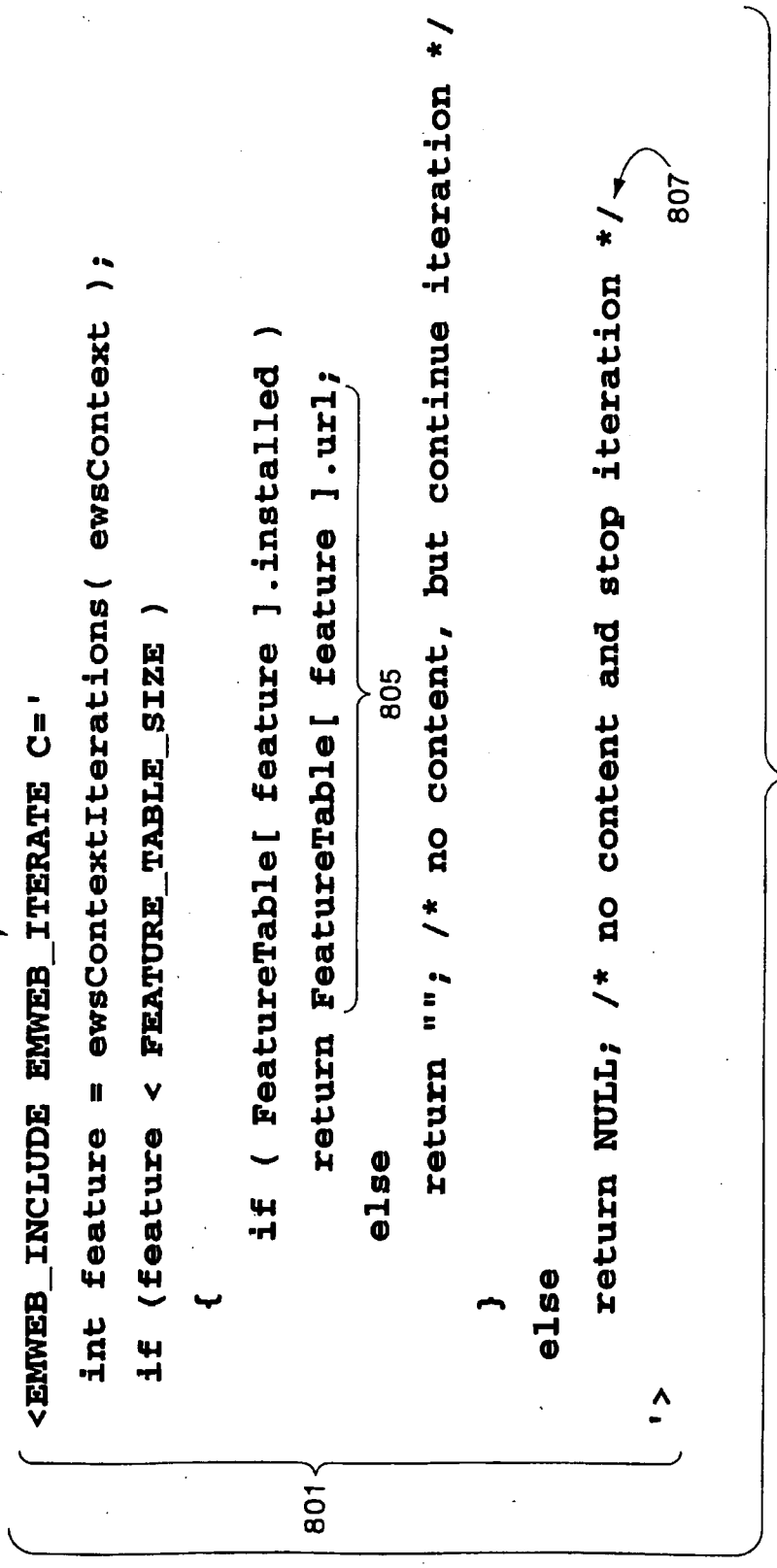
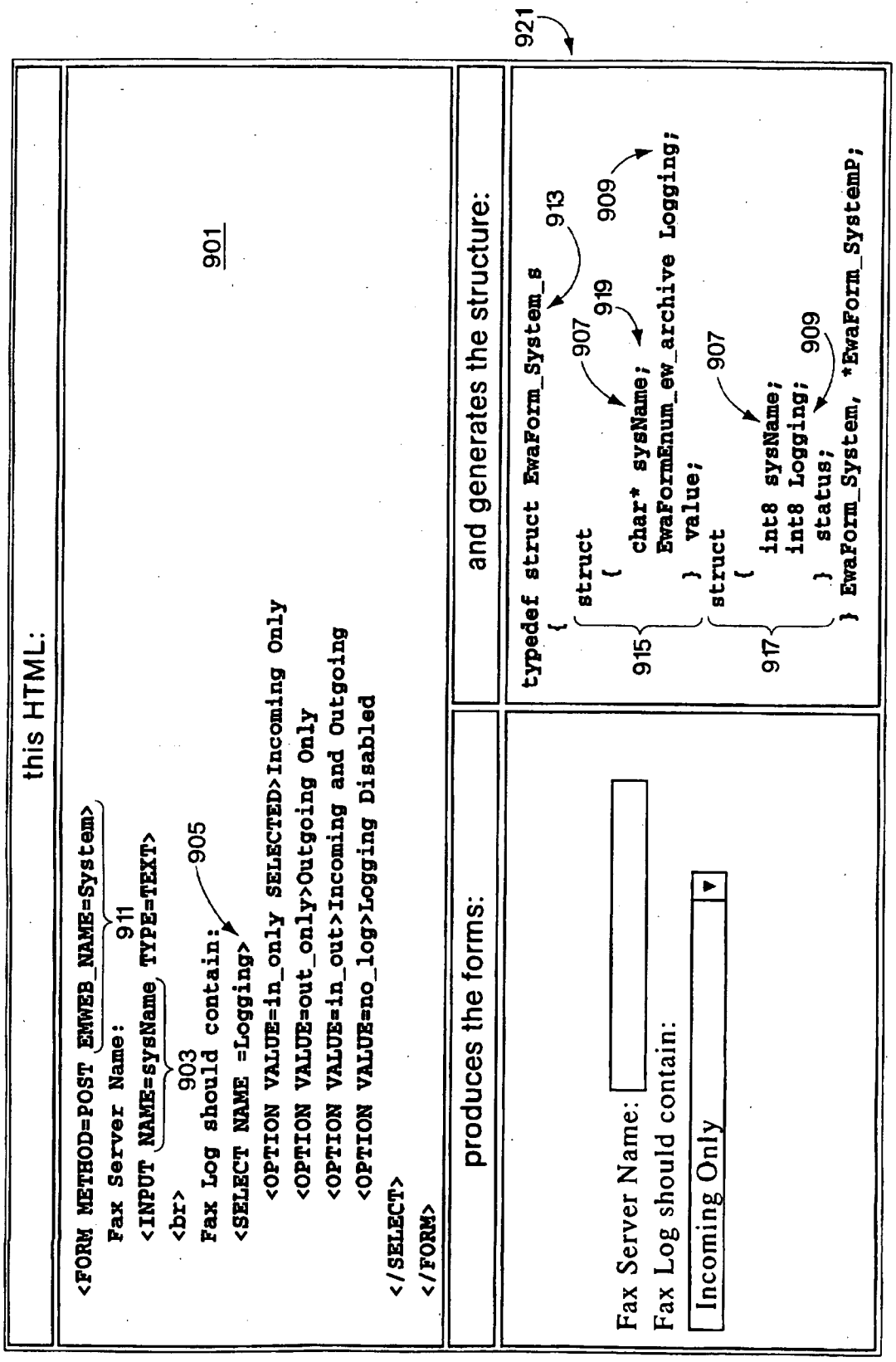

<EMWEB_INCLUDE EMWEB_ITERATE C='
int feature = ewsContextIterations(ewsContext);
if (feature < FEATURE_TABLE_SIZE)
{
if (FeatureTable[feature].installed)
return FeatureTable[feature].url;
else
return ""; /* no content, but continue iteration */
}
else
return NULL; /* no content and stop iteration */
'>
803
801
805
807

Fig. 8


this HTML:
<FORM METHOD=POST EMWEB_NAME=System>
Fax Server Name:
<INPUT NAME=sysName TYPE=TEXT>

Fax Log should contain:
<SELECT NAME =Logging>
<OPTION VALUE=in_only SELECTED>Incoming Only
<OPTION VALUE=out_only>Outgoing Only
<OPTION VALUE=in_out>Incoming and Outgoing
<OPTION VALUE=no_log>Logging Disabled
</SELECT>
</FORM>
901
903
905
911
produces the forms:
and generates the structure:
921
typedef struct EwaForm_System_s
{
struct
{
char* sysName;
EwaFormEnum_ew_archive Logging;
} value;
} 915
struct
{
int8 sysName;
int8 Logging;
} status;
} 917
EwaForm_System, *EwaForm_SystemP;
907
907
909
909
913
919
Fax Server Name:
Fax Log should contain:
Incoming Only

Fig. 9

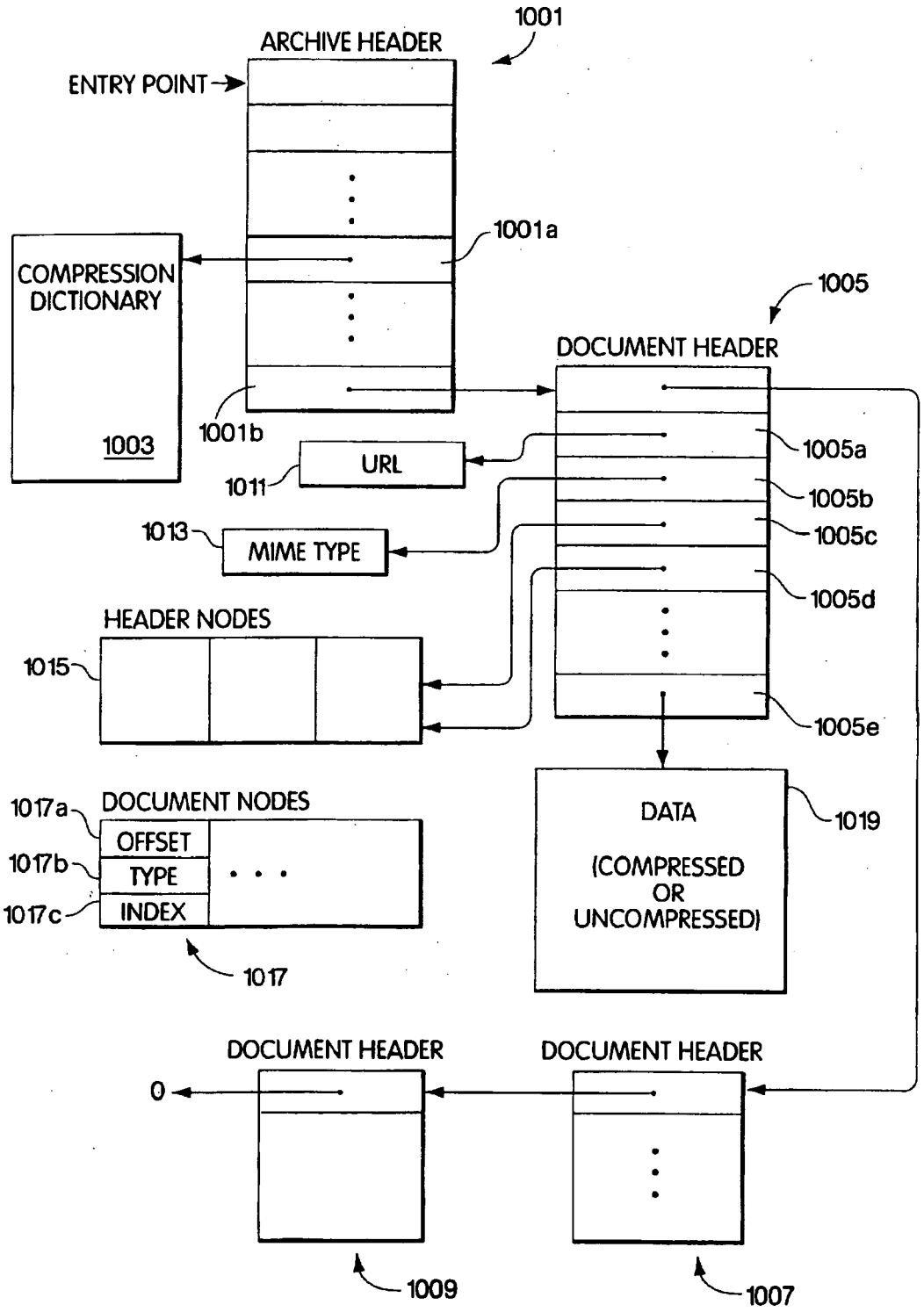

1001
ARCHIVE HEADER
ENTRY POINT
1001a
COMPRESSION DICTIONARY
1003
1001b
1005
DOCUMENT HEADER
1005a
1005b
1005c
1005d
1005e
URL
1011
1013
MIME TYPE
HEADER NODES
1015
1017a
DOCUMENT NODES
OFFSET
1017b
TYPE
1017c
INDEX
1017
DATA
(COMPRESSED OR UNCOMPRESSED)
1019
DOCUMENT HEADER
DOCUMENT HEADER
0
1009
1007

Fig. 10

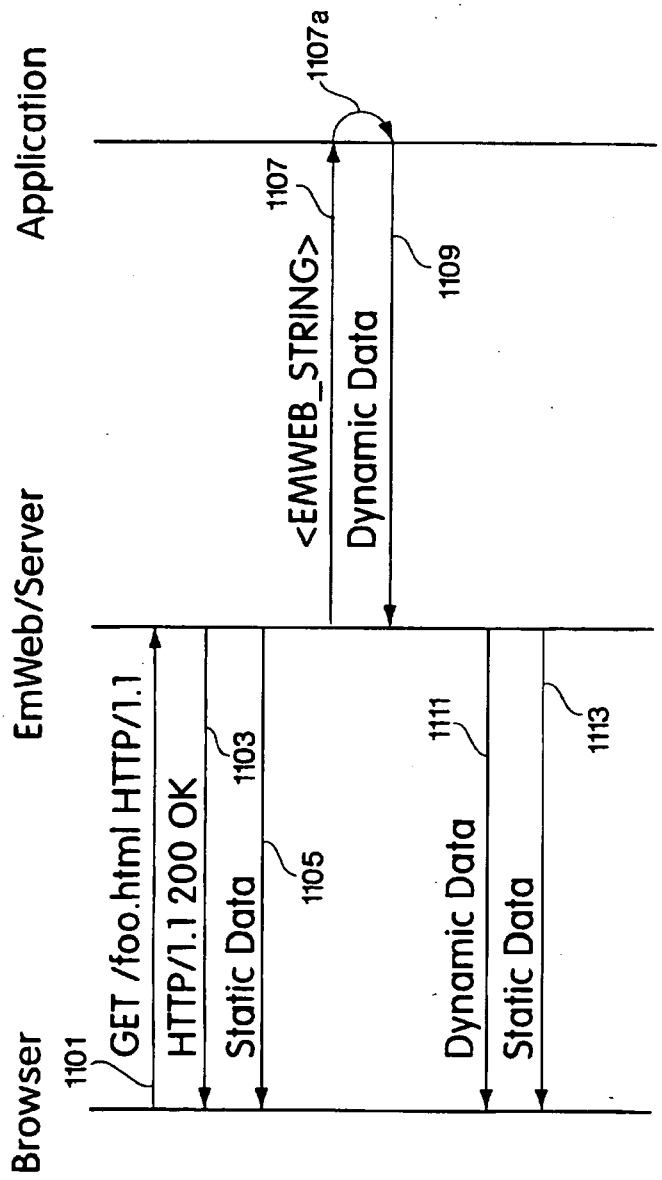

Application
EmWeb/Server
Browser
1107a
1107
<EMWEB_STRING>
Dynamic Data
1109
1101
GET /foo.html HTTP/1.1
HTTP/1.1 200 OK
1103
Static Data
1105
1111
Dynamic Data
1113
Static Data

Fig. 11

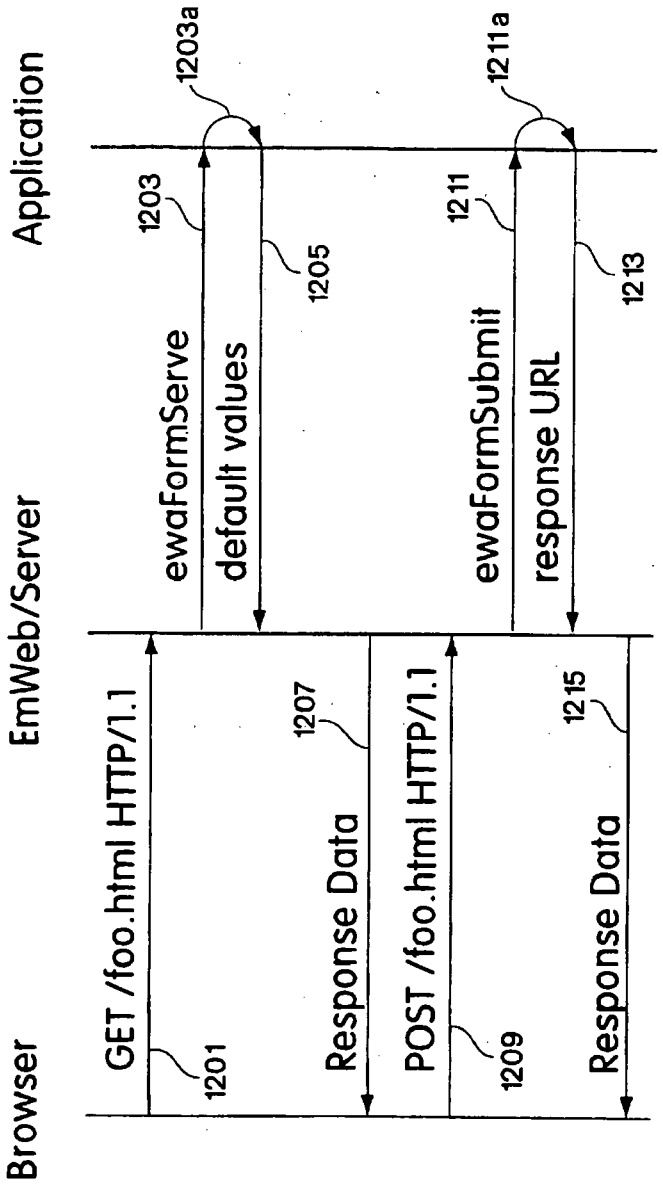

Browser
EmWeb/Server
Application
GET /foo.html HTTP/1.1
1201
Response Data
1207
POST /foo.html HTTP/1.1
1209
Response Data
1215
ewaFormServe
1203
1203a
default values
1205
ewaFormSubmit
1211
1211a
response URL
1213

Fig. 12

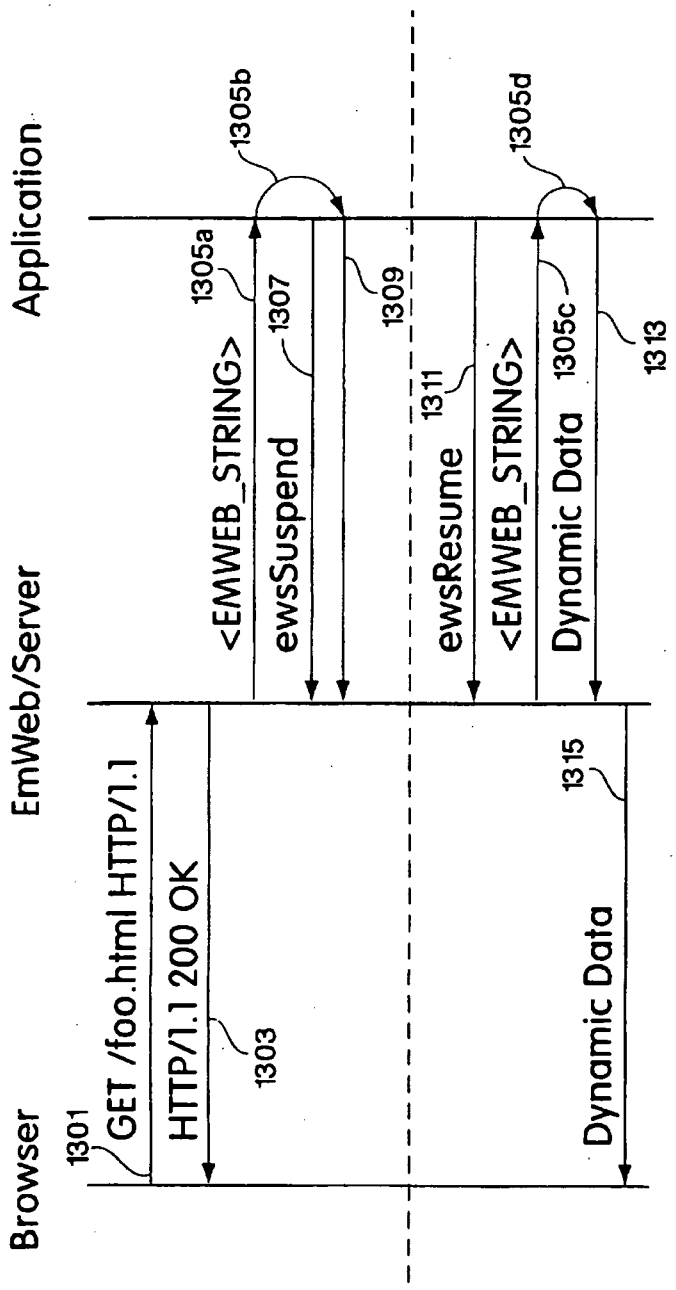

Browser
EmWeb/Server
Application
GET /foo.html HTTP/1.1
1301
HTTP/1.1 200 OK
1303
<EMWEB_STRING>
1305a
ewsSuspend
1307
1309
ewsResume
1311
<EMWEB_STRING>
1305b
Dynamic Data
1305c
1313
Dynamic Data
1315
1305d

Fig. 13

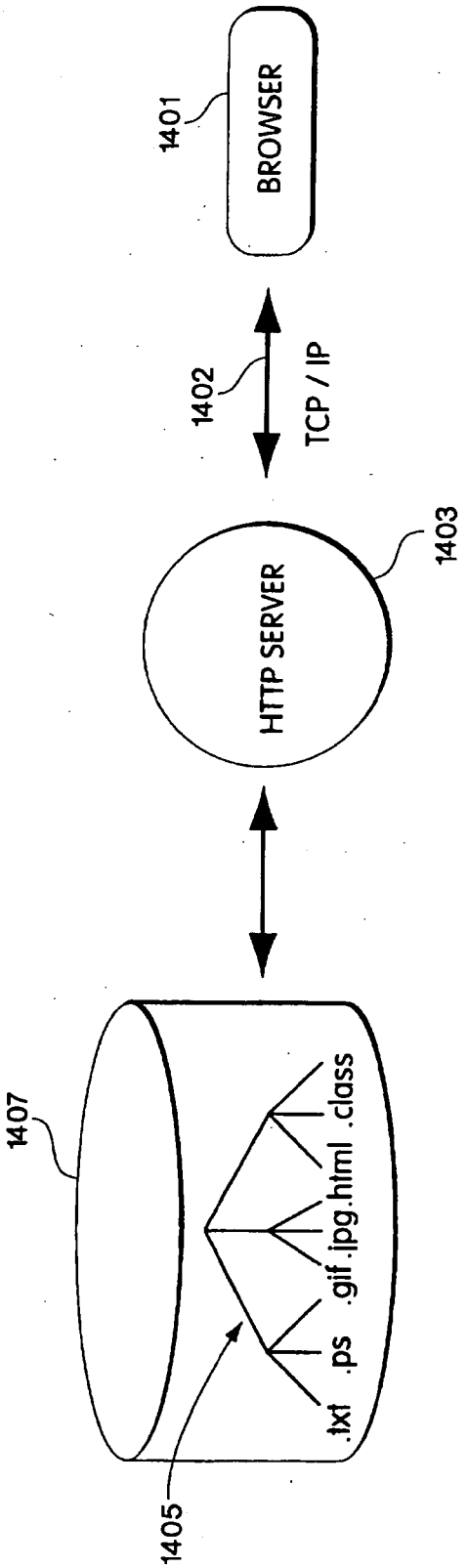

1401
BROWSER
1402
TCP / IP
1403
HTTP SERVER
1407
.txt .ps .gif .jpg .html .class
1405

Fig. 14

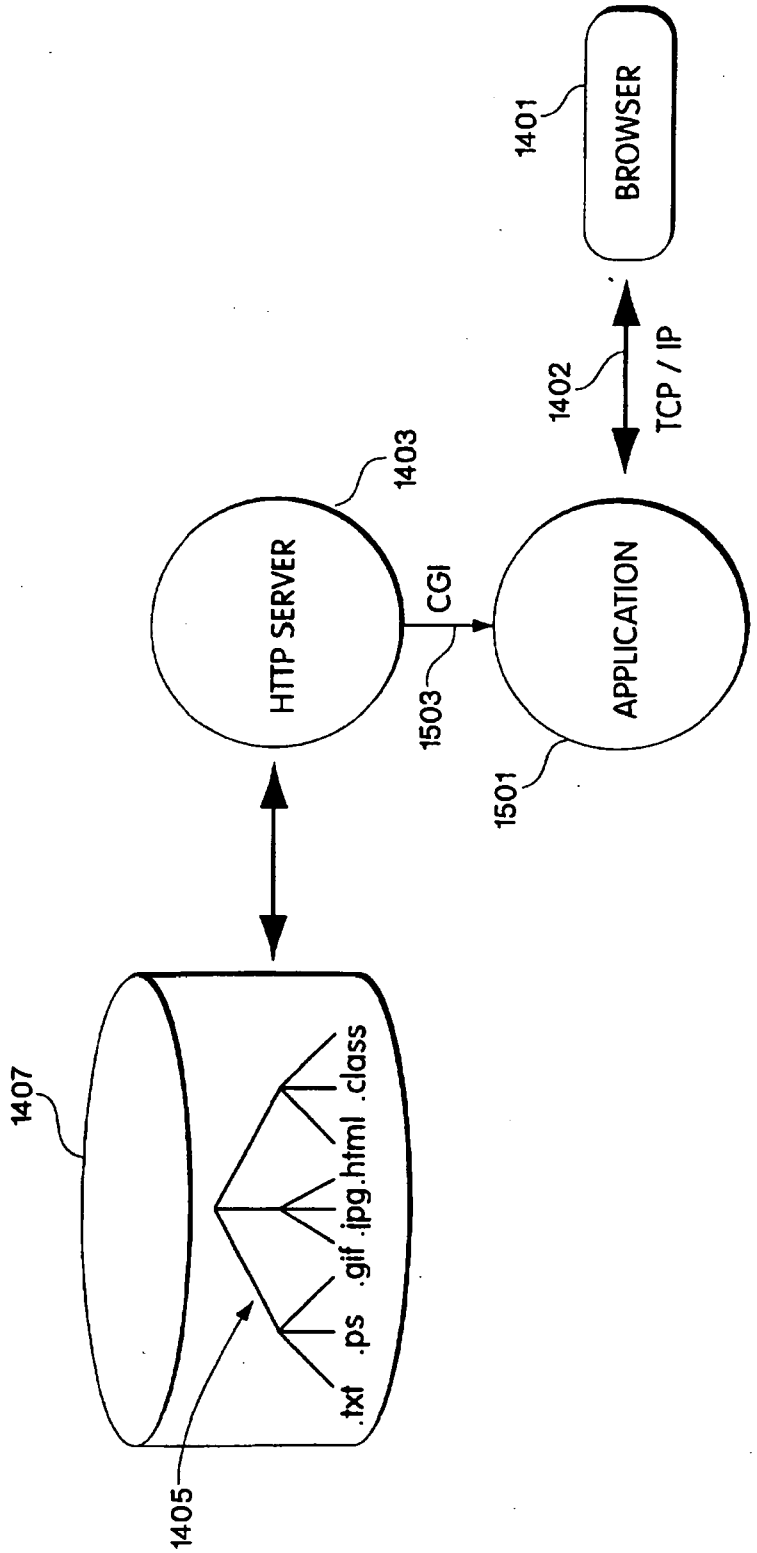

1401
BROWSER
1402
TCP / IP
1403
HTTP SERVER
CGI
1503
APPLICATION
1501
1407
.class
.html
.jpg
.gif
.ps
.txt
1405

Fig. 15

INTERNATIONAL SEARCH REPORT

In ational Application No
PCT/US 97/13817

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F9/44 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CRESPO A ET AL: "WebWriter: A browser-based editor for constructing Web applications" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, page 1291-1306 XP004018228 see page 1393, left-hand column, line 32 - right-hand column, line 35 see page 1296, left-hand column, line 46 - page 1297, right-hand column, line 11 see page 1302, right-hand column, line 10 - page 1303, left-hand column, last line --- -/--	1-20

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

3

Date of the actual completion of the international search	Date of mailing of the international search report
11 December 1997	30/12/1997

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Fonderson, A

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int'l Application No
PCT/US 97/13817

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	D. A. LADD AND J. C. RAMMING: "Programming the Web: An Application-Oriented Language for Hypermedia Service Programming" PROCEEDINGS OF THE 4TH INTERNATIONAL WORLD WIDE WEB CONFERENCE, December 1995, BOSTON, MA, USA, XP002049893 (http://www.w3.org/pub/Conferences/WWW4/Papers/251/) see abstract see page 2, line 3 - line 8 see page 10, line 51 - page 16, last line	1-20

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

PCT ORGANISATION MONDIALE DE LA PROPRIETE INTELLECTUELLE
Bureau international

DEMANDE INTERNATIONALE PUBLIEE EN VERTU DU TRAITE DE COOPERATION EN MATIERE DE BREVETS (PCT)

(51) Classification internationale des brevets 6 :		(11) Numéro de publication internationale:
H04M 3/50	A1	WO 98/59486
		(43) Date de publication internationale: 30 décembre 1998 (30.12.98)

(21) Numéro de la demande internationale: PCT/CH97/00246

(22) Date de dépôt international: 20 juin 1997 (20.06.97)

(71) Déposant *(pour tous les Etats désignés sauf US)*: TELECOM PTT [CH/CH]; Forschung + Entwicklung, FE, Ostermundigenstrasse 93, CH–3000 Berne 29 (CH).

(72) Inventeur; et
(75) Inventeur/Déposant *(US seulement)*: VAN KOMMER, Robert [NL/CH]; Route du Coteau 45, CH–1752 Villars–sur–Glâne (CH).

(74) Mandataire: BOVARD S.A.; Optingenstrasse 16, CH–3000 Berne 25 (CH).

(81) Etats désignés: AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GE, GH, HU, IL, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, US, UZ, VN, YU, ZW, brevet ARIPO (GH, KE, LS, MW, SD, SZ, UG), brevet eurasien (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), brevet européen (AT, BE, CH, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), brevet OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, ML, MR, NE, SN, TD, TG).

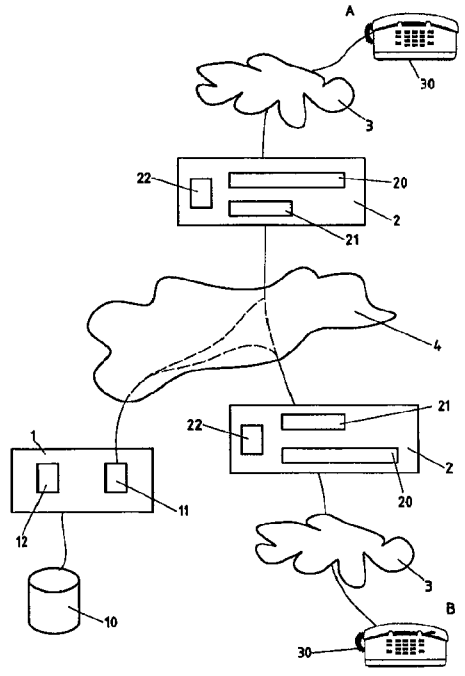
Publiée
Avec rapport de recherche internationale.

(54) **Title:** SYSTEM AND METHOD FOR CODING AND BROADCASTING VOICE DATA

(54) **Titre:** SYSTEME ET PROCEDE DE CODAGE ET DE DIFFUSION D'INFORMATIONS VOCALES

(57) Abstract

The invention concerns a system for broadcasting voice data through a packet switching telecommunication network, from at least one platform for an automatic voice teleservice system (1). The messages broadcast by the platform, for example meteorological forecasts, can be listened to from a plurality of terminals, for example telephones (30). The messages are transmitted through the telecommunication network in coded form, the code including a semantic representation of the messages, then converted into audio signals at the (2) access point to said packet transmission telecommunication network. The conversion is, for example, carried out by means of a voice synthesis module, such as a TTS (text–to–speech) module. The network (2) access points are geographically distributed and can generally be accessed by a local tariff call. The packet transmission telecommunication network is for example a ATM network or internet. The system is also adapted to interactive teleservice systems (IVR, interactive voice response systems).



(57) Abrégé

Système de diffusion d'informations vocales à travers un réseau de télécommunication à commutation de paquets, à partir d'au moins une plate–forme pour système de téléservice vocal automatique (1). Les messages diffusés par la plate–forme, par exemple des prévisions météorologiques, peuvent être écoutés depuis une pluralité de terminaux, par exemple de téléphones (30). Les messages sont transmis à travers le réseau de télécommunication sous forme codée, le code incluant une représentation sémantique des messages, puis convertis en signaux audio au niveau d'un point d'accès (2) audit réseau de télécommunication à transmission de paquets. La conversion est par exemple effectuée au moyen d'un module de synthèse vocale, par exemple d'un module TTS (text–to–speech). Les points d'accès réseau (2) sont distribués géographiquement et peuvent généralement être accédés par un appel en tarif local. Le réseau de télécommunication à transmission de paquets est par exemple un réseau ATM ou internet. Le système est également adapté à des systèmes de téléservice interactifs (IVR, interactive voice response systems).

UNIQUEMENT A TITRE D'INFORMATION

Codes utilisés pour identifier les Etats parties au PCT, sur les pages de couverture des brochures publiant des demandes internationales en vertu du PCT.

AL	Albanie	**ES**	Espagne	**LS**	Lesotho	**SI**	Slovénie
AM	Arménie	**FI**	Finlande	**LT**	Lituanie	**SK**	Slovaquie
AT	Autriche	**FR**	France	**LU**	Luxembourg	**SN**	Sénégal
AU	Australie	**GA**	Gabon	**LV**	Lettonie	**SZ**	Swaziland
AZ	Azerbaïdjan	**GB**	Royaume-Uni	**MC**	Monaco	**TD**	Tchad
BA	Bosnie-Herzégovine	**GE**	Géorgie	**MD**	République de Moldova	**TG**	Togo
BB	Barbade	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tadjikistan
BE	Belgique	**GN**	Guinée	**MK**	Ex-République yougoslave de Macédoine	**TM**	Turkménistan
BF	Burkina Faso	**GR**	Grèce			**TR**	Turquie
BG	Bulgarie	**HU**	Hongrie	**ML**	Mali	**TT**	Trinité-et-Tobago
BJ	Bénin	**IE**	Irlande	**MN**	Mongolie	**UA**	Ukraine
BR	Brésil	**IL**	Israël	**MR**	Mauritanie	**UG**	Ouganda
BY	Bélarus	**IS**	Islande	**MW**	Malawi	**US**	Etats-Unis d'Amérique
CA	Canada	**IT**	Italie	**MX**	Mexique	**UZ**	Ouzbékistan
CF	République centrafricaine	**JP**	Japon	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Pays-Bas	**YU**	Yougoslavie
CH	Suisse	**KG**	Kirghizistan	**NO**	Norvège	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	République populaire démocratique de Corée	**NZ**	Nouvelle-Zélande		
CM	Cameroun			**PL**	Pologne		
CN	Chine	**KR**	République de Corée	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Roumanie		
CZ	République tchèque	**LC**	Sainte-Lucie	**RU**	Fédération de Russie		
DE	Allemagne	**LI**	Liechtenstein	**SD**	Soudan		
DK	Danemark	**LK**	Sri Lanka	**SE**	Suède		
EE	Estonie	**LR**	Libéria	**SG**	Singapour		

SYSTEME ET PROCEDE DE CODAGE ET DE DIFFUSION D'INFORMATIONS VOCALES

La présente invention concerne un système et un procédé de diffusion d'informations vocales. Plus particulièrement, la présente invention concerne un système et un procédé de téléservices vocaux, permettant à un utilisateur d'accéder au moyen d'un téléphone, ou de tout type de dispositif muni de moyens d'émission acoustiques, à des informations diffusées par une plate-forme centralisée dans un réseau de télécommunication.

En dépit de l'émergence d'autres systèmes de diffusion automatique d'informations, par exemple internet ou télétexte, l'offre de téléservices vocaux tend à se développer. Ces services peuvent en effet être utilisés sans équipements particuliers, si ce n'est un combiné téléphonique, et peuvent donc être consultés aisément depuis n'importe où, par exemple au moyen d'un téléphone portable. En outre, l'accès à l'information est possible sans connaissances techniques particulières. Des exemples de téléservices vocaux populaires comprennent par exemple l'horloge parlante, la diffusion des prévisions météorologiques, des résultats sportifs, des cours de la bourse, etc...

Les informations diffusées sont généralement mémorisées sous forme numérique ou enregistrées sur un support d'enregistrement audio dans une plate-forme centralisée et reliée au réseau téléphonique. L'utilisateur prend l'initiative de consulter ces informations en composant sur son combiné téléphonique le numéro de téléphone de la plate-forme. Une connexion téléphonique ordinaire est alors établie entre la plate-forme centralisée et l'utilisateur, et un dialogue vocal est effectué via cette connexion. La connexion est bidirectionnelle, même lorsque l'information est diffusée dans une seule direction (c'est typiquement le cas pour un service d'horloge parlante) ou majoritairement dans une direction (par exemple dans les systèmes où les seules réponses attendues de l'utilisateur se limitent à l'introduction d'un mot de passe ou de réponses brèves du type oui-non). La charge occasionnée sur le réseau de télécommunication est donc sans mesure avec le volume effectif d'informations utiles transmis.

Les progrès observés depuis quelques années dans le domaine du traitement de la voix, en particulier de la synthèse de voix (systèmes TTS - « text-to-speech »), de l'analyse de voix et de la reconnaissance vocale de personne, ouvrent tout grand la porte à de nouveaux systèmes de téléservices vocaux, étendant les possibilités offertes par les systèmes conventionnels. Le domaine d'application de ces nouvelles technologies n'est plus limité à des services de diffusion unidirectionnelle d'information (comme l'horloge parlante), mais concerne également des services d'information interactifs, offrant des possibilités de dialogue entre l'appelant et la plate-forme de téléservice vocal. Ces systèmes interactifs sont généralement connus sous le sigle anglophone IVR (Interactif Voice Response Systems) et appliqués notamment pour des services de télébanking. On connaît par exemple déjà des systèmes offrant à l'utilisateur la possibilité de choisir, au moyen de commandes vocales, les informations qu'il désire écouter, voire de modifier ces informations ou d'initier le lancement de programmes d'applications exécutées par la plate-forme de téléservice vocal. A titre d'exemple, le document de brevet WO88/05239 décrit un système permettant d'effectuer des sondages ou des votations de manière automatique. WO93/26113 décrit un autre système de messagerie vocale largement automatisé.

Les systèmes de téléservice vocal sont généralement opérés au moyen d'une plate-forme centralisée, pour des raisons évidentes de mise à jour des informations et de coût. La longueur de la connexion téléphonique établie dépend donc de la distance entre le point d'appel de l'utilisateur et l'emplacement de la plate-forme. L'attrait d'un téléservice vocal diminue cependant fortement pour les abonnés qui ne peuvent pas l'appeler en tarif local. Dans de nombreux cas, le gestionnaire du réseau de télécommunication propose donc un tarif d'appel unique pour un téléservice donné, qui pénalise les abonnés les plus proches et est déficitaire pour les appels d'abonnés domiciliés plus loin.

Le document de brevet EP-A2-0559981 décrit un système de téléservice vocal interactif dans lequel l'utilisateur est connecté à la plate-forme centralisée au moyen d'une connexion entièrement numérique. Le but de ce système est essentiellement de limiter le nombre de conversions analogiques-

numériques et numériques-analogiques entre l'utilisateur et la plate-forme de téléservice vocal. Les problèmes mentionnés ci-dessus ne sont cependant pas résolus par ce système.

Un but de la présente invention est donc de proposer un système de téléservice vocal amélioré. Plus particulièrement, un but de la présente invention est de proposer un service de téléservice vocal occasionnant une charge moindre du réseau de télécommunication.

Ces buts sont atteints notamment au moyen des éléments des parties caractérisantes des revendications indépendantes, des modes de réalisation préférentiels étant indiqués dans les revendications dépendantes.

L'invention part de la constatation que, dans l'art antérieur, les messages vocaux échangés, notamment les informations vocales diffusées par la plate-forme vers l'appelant, nécessitent une largeur de bande beaucoup plus importante que le minimum requis pour transmettre le seul contenu sémantique de l'information transmise.

Les buts de l'invention sont donc atteints en transmettant, non pas des signaux audio analogiques ou numériques (phonétiques), mais des messages vocaux contenant uniquement une information sur la sémantique de ces signaux.

Dans le cas d'un système numérique, la transmission de milliers d'échantillons de voix successifs, codés par exemple sur 8 bits, est ainsi remplacée par la transmission de quelques caractères, par exemple de quelques codes ASCII correspondant à la représentation en mode texte ou pseudo-texte du contenu sémantique des messages.

Le taux de compression ainsi obtenu, et donc la charge du réseau, est très élevé, typiquement de l'ordre de 1 :500 par exemple, selon les messages et le type d'application. La conversion du message sémantique en signal audio est effectuée de préférence par un point d'accès réseau à proximité de l'utilisateur, en sorte qu'un signal audio à grande largeur de bande

est transmis seulement entre le point d'accès réseau et l'utilisateur, soit sur une fraction de la distance totale séparant l'utilisateur de la plate-forme de téléservice.

Etant donné le débit d'information très faible obtenu grâce au procédé de l'invention, et la nature irrégulière de ce débit, notamment dans le cas de systèmes interactifs, les messages codés sous forme sémantique sont avantageusement transmis sous forme de paquets entre la plate-forme centralisée et le ou les points d'accès réseau. Avantageusement, le réseau de télécommunication reliant la plate-forme centralisée aux points d'accès réseau est alors constitué par un réseau à transmission de paquets, par exemple un réseau au protocole ATM et/ou internet (TCP/IP). Dans ce dernier cas, les points d'accès réseau peuvent avantageusement être constitués par des POPs (Points of Presence) distribués du réseau internet.

L'utilisateur souhaitant recourir à un système de téléservice vocal, par exemple écouter les prévisions météorologiques, appelle au moyen de son téléphone le POP le plus proche. Cette communication peut généralement être établie au tarif local. Le POP contacte alors la plate-forme centralisée gérant le système de téléservice vocal automatique, qui lui transmet un message codé incluant une représentation sémantique contenant les informations requises, par exemple un fichier texte annonçant les prévisions météorologiques pour la région considérée. Ce fichier texte est converti en signal audio (analogique ou numérique) au moyen d'un dispositif de synthèse de voix dans le POP (dispositif TTS, « Text-To-Speech »), et le signal audio obtenu est transmis à l'utilisateur qui peut alors l'écouter.

Une connexion téléphonique ordinaire est seulement requise entre l'utilisateur et le point d'accès réseau (POP). Etant donné la forte concentration de points d'accès au réseau internet, cette connexion peut très souvent être établie en bénéficiant du tarif local. Entre le POP et la plate-forme, la connexion est avantageusement de type internet, et son coût ne dépend donc pas de la distance. En outre, le volume d'informations échangé entre le POP et la plate-forme est très fortement réduit grâce à la conversion sous forme sémantique des messages vocaux.

Au cas où un temps de réponse court est nécessaire, par exemple en cas de communication interactive, la paquetisation des messages dans Internet peut causer un retard d'autant plus grand que la compression est importante. Un paquetiseur n'expédie en effet généralement que des paquets complets. Une solution permettant de réduire le délai est d'insérer des octets de bourrage après le message utile, permettant de remplir les paquets et donc de les transmettre immédiatement. Une solution préférentielle consiste à multiplexer plusieurs messages à l'intérieur de chaque paquet, puis à les démultiplexer à l'intérieur du réseau de télécommunication.

Avantageusement, les points d'accès réseau distribués contiennent une mémoire de type cache qui mémorise temporairement les messages transmis depuis la plate-forme centralisée. Une connexion avec la plate-forme n'est ainsi pas requise à chaque interrogation par un utilisateur, ce qui contribue à réduire encore la charge à travers le réseau de télécommunication. La mémoire de type cache peut soit mémoriser les messages codés sous forme sémantique, ce qui permet de réduire considérablement la taille de mémoire requise, soit les signaux audio synthétisés pour limiter le nombre de conversions nécessaires.

L'invention sera mieux comprise à l'aide de la description donnée à titre d'exemple non limitatif et illustrée par la figure unique qui montre sous forme schématique un système complet de diffusion d'informations vocales à travers un réseau de télécommunication.

La description concerne principalement une application de l'invention à un système de téléservice vocal, par exemple un service de diffusion de prévisions météorologiques accessible par téléphone. Il est important néanmoins de voir que l'invention peut aussi être appliquée à n'importe quel type de système de diffusion d'informations vocales monodirectionnel, bidirectionnel ou interactif.

Sur la figure 1, le numéro de référence 1 désigne une plate-forme centralisée pour système de téléservice vocal. La plate-forme 1 fournit l'information requise et, dans le cas d'un système de téléservice interactif, gère

le dialogue avec l'utilisateur. La plate-forme 1 contient de préférence une mémoire 10 mémorisant l'information à diffuser. Seul le contenu sémantique de l'information est mémorisé, par exemple sous forme de fichier texte, pseudo-texte ou sous n'importe quel format approprié. Des marqueurs prosodiques, ou d'autres types d'informations permettant d'améliorer la qualité de la voix synthétisée à partir de cette information, sont de préférence inclus dans ce fichier. Dans un autre mode de réalisation, la mémoire 10 contient des données audio, enregistrées par exemple sur une bande magnétique ou sur tout type de support analogique ou numérique approprié, qui sont converties à chaque lecture par des moyens d'analyse de voix en données codées incluant une représentation sémantique. Dans un troisième mode de réalisation, l'information diffusée par la plate-forme 1 n'est pas préparée en avance, mais générée par un système de synthèse de messages en fonction par exemple des réponses fournies par l'utilisateur dans le cas d'un téléservice interactif IVR.

La plate-forme 1 contient en outre de préférence des moyens d'analyse de voix, permettant d'analyser les réponses de l'utilisateur dans le cas d'un système interactif. La plate-forme 1 peut en outre comprendre des moyens de traitement aptes à exécuter n'importe quel type d'application appropriée selon le type de téléservice et les commandes données par l'utilisateur, par exemple débiter un compte bancaire dans le cas d'un service de télébanking interactif, ou enregistrer une réponse de l'utilisateur. Les moyens de gestion de dialogue, d'analyse de voix, de génération de messages et de traitement sont de préférence implémentés sous forme de modules informatiques pouvant être exécutés par des moyens de traitement 12. Ces modules ou programmes peuvent par exemple être stockés sur un support de données informatique, tel que disque dur, disquette ou cd-rom par exemple, pouvant être lu et exécuté par un dispositif programmable constituant la plate-forme 1. La plate-forme 1 contient en outre une interface 11 permettant de la connecter au réseau de télécommunication 4, dans ce cas un réseau à transmission de paquets de type internet. La plate-forme 1 constitue donc dans cet exemple un serveur d'information internet.

Le système de l'invention comporte en outre une pluralité de points d'accès réseau distribués 2. Seuls deux points d'accès réseau 2 sont représentés sur la figure 1 ; le nombre de points d'accès réseau peut cependant être quelconque et dépend principalement de la taille du réseau de télécommunication 4. Dans cet exemple, les points d'accès réseau sont constitués par des points de présence POP dans le réseau internet. Les points d'accès réseau 2 comprennent de manière générale une interface 21, permettant de les connecter au réseau 4, des moyens de traitement 20, ainsi qu'une mémoire cache 22 dont la fonction sera décrite plus bas. Les points d'accès réseau 2 comprennent en outre des moyens d'interface non représentés, par exemple des moyens de type modem, avec un réseau d'accès 3, par exemple avec un réseau de téléphonie fixe ou mobile conventionnel ou avec un réseau RNIS ou CATV.

Les moyens de traitement 20 comprennent des moyens de synthèse de voix, implémentés par exemple sous la forme d'un module informatique exécutable par un processeur universel. Les moyens de synthèse de voix permettent de convertir les messages vocaux codés sous forme sémantique en signaux audio phonétiques. Différents algorithmes et programmes commerciaux, désignés de manière générale sous le nom d'algorithmes TTS (Text-To-Speech), sont connus à cet effet dans l'art antérieur. Il va de soi que l'algorithme de synthèse de voix utilisé doit être compatible avec le codage sémantique adopté par la plate-forme centralisée 1, et être par exemple apte le cas échéant à reconnaître des marqueurs prosodiques, ou d'autres informations caractérisant la voix, insérés dans le fichier texte. Avantageusement, l'algorithme de synthèse de voix peut donc être téléchargé depuis la plate-forme centralisée 1 à travers le réseau internet. Il est ainsi possible de mettre à jour facilement tous les points d'accès 2 lors de modifications de l'algorithme de codage sémantique utilisé par la plate-forme 1.

Le point d'accès réseau 2 est avantageusement constitué par un ordinateur, par exemple par un ordinateur polyvalent ou par un serveur de communication spécialisé, exécutant un programme informatique approprié mémorisé sur un support de donnée informatique adéquat.

La mémoire cache 22, qui peut être gérée par n'importe quel algorithme connu, mémorise temporairement les messages vocaux transmis depuis la plate-forme centralisée 1 via le réseau 4. La mémoire de type cache peut soit mémoriser la représentation sémantique des messages codés, ce qui permet de réduire considérablement la taille de mémoire requise, soit les signaux audio synthétisés pour limiter le nombre de conversions nécessaires. La mémoire cache peut par exemple être mise à jour à intervalles réguliers, par exemple 4 fois par jour dans le cas d'un téléservice d'informations météorologiques, ou après un nombre prédéfini de consultations par les utilisateurs, ou seulement lors d'une consultation par l'utilisateur, lorsqu'il s'avère que les données dans la mémoire cache sont trop anciennes pour être fiables.

Le procédé de l'invention, exécuté par le dispositif de la figure 1, va maintenant être expliqué à l'aide d'un exemple simple de téléservice diffusant des informations météorologiques. Il est bien entendu qu'il s'agit là d'une application donnée à titre d'exemple non limitatif, permettant de comprendre aisément le procédé de l'invention, mais que l'invention peut s'appliquer également à tout type de système de diffusion d'informations vocales.

L'utilisateur désirant écouter les prévisions météorologiques au moyen d'un système de téléservice selon l'invention compose sur son terminal 30 le numéro d'appel du service considéré, en Suisse par exemple le numéro 162. Une communication est alors établie à travers le réseau d'accès conventionnel 3 avec le point d'accès réseau 2 le plus proche, par exemple avec le fournisseur d'accès internet le plus proche. Selon la densité de points d'accès réseau 2, cette connexion peut être établie au tarif local, donc pour un coût très modique. L'utilisateur et le fournisseur du téléservice évitent donc de devoir se partager le coût d'une communication téléphonique entre le terminal 30 et le dispositif 1 généralement plus éloigné.

Après établissement de la communication, les moyens de traitement 20 dans le point d'accès réseau 2 vérifient si les informations requises, dans ce cas les prévisions météorologiques, sont contenues dans la mémoire cache 22. Si c'est le cas, et si les données mémorisées sont fiables, le contenu de la

mémoire cache est lu et ces informations peuvent être écoutées par l'utilisateur. Si la mémoire cache 22 contient des informations déjà synthétisées sous forme de signal sonore, ce signal peut directement être émis au moyen de l'interface non représentée avec le réseau d'accès 3 et écouté au moyen du haut-parleur dans le dispositif 30. Si en revanche la mémoire cache 22 contient uniquement la représentation sémantique des messages codés, par exemple un fichier texte avec éventuellement des marqueurs prosodiques, ce fichier est converti par les moyens de traitement 20 en signal audio au moyen d'un module de synthèse de voix TTS, puis transmis au dispositif 30.

Lorsque la mémoire cache 22 ne contient pas les informations à jour requises par l'utilisateur, ou si le dispositif 2 ne comporte pas de mémoire cache, le point d'accès réseau envoie une requête à la plate-forme 1 via le réseau de transmission de paquets 4. Dans le cas d'un réseau internet, cette requête peut par exemple être constituée par un message internet adressé à la plate-forme 1.

Dans le cas d'un téléservice monodirectionnel, c'est-à-dire d'un téléservice dans lequel l'information circule uniquement depuis la plate-forme 1 vers l'utilisateur, la plate-forme 1 génère alors l'information requise et l'envoie au point d'accès 2 choisi sous forme de message vocal codé incluant une représentation sémantique de l'information. Le message vocal envoyé par la plate-forme 1 peut dans cet exemple être constitué par un simple fichier en mode texte ou pseudo-texte indiquant les prévisions météorologiques demandées. Tout autre type de codage sémantique, par exemple un codage avec un alphabet phonétique, ou un codage utilisant des marqueurs prosodiques, peut cependant être utilisé dans le cadre de l'invention.

Le volume d'information transmis est très faible, en raison du codage sémantique, et peut donc être transmis très rapidement à travers le réseau de télécommunication 4. Le point d'accès réseau 2 convertit ensuite le message codé sous forme sémantique en signal audio grâce au moyens de synthèse de voix évoqués, et transmet comme ci-dessus ce signal audio à l'utilisateur à travers le réseau d'accès 3. L'utilisateur peut alors écouter ce signal audio au moyen du haut-parleur du dispositif 30.

Dans le cas d'un téléservice bidirectionnel, par exemple d'un téléservice interactif permettant à l'utilisateur d'influencer le fonctionnement de la plate-forme 1, la plate-forme 1 peut envoyer une question à l'utilisateur, par exemple lui demander quelles informations exactement il désire écouter. Par exemple, la plate-forme 1 peut envoyer un message vocal à l'utilisateur 1 lui demandant le nom de la région à laquelle doit se rapporter le bulletin météorologique. Ce message vocal est transmis sous forme sémantique vers la plate-forme 2, puis converti de la manière indiquée en un signal audio qui peut être écouté par l'utilisateur du terminal 30. L'utilisateur peut alors répondre, soit au moyen du clavier du dispositif 30, soit de préférence vocalement.

Dans le cas d'une réponse vocale, le point d'accès réseau 2 convertit, grâce à des moyens d'analyse de voix réalisés sous la forme d'un module informatique exécutable par les moyens de traitement 20, cette réponse en un message codé sous forme sémantique, et transmet cette réponse à la plate-forme 1. Les moyens de traitement adoptent alors un comportement dépendant de la réponse, et exécutent une procédure appropriée dépendant de l'application. Dans le cas d'un téléservice de diffusion de bulletins météorologiques, la procédure peut par exemple consister en la diffusion d'un bulletin correspondant à la région sélectionnée.

Ce bulletin est alors transmis de la manière décrite, codé sous forme sémantique et converti en signal audio par le point d'accès 2 afin de pouvoir être écouté par l'utilisateur. Selon l'algorithme de gestion choisi pour le cache 22, ce message peut également être copié dans ce cache.

Dans le cas le plus simple, le format des messages codés sous forme sémantique est simplement de type texte. Par exemple, des simples fichiers ASCII peuvent être échangés entre la plate-forme 1 et les points d'accès réseau 2. Un fichier texte ne contient toutefois aucun paramètres caractérisant la voix du lecteur, et ne permet pas de synthétiser aisément une voix naturelle. Pour améliorer la synthèse, il est préférable de transmettre un fichier pseudo-texte contenant des marqueurs prosodiques supplémentaires en sus du texte.

Il est naturellement possible de transmettre seulement une partie du message codé sous forme sémantique, et une autre partie, ou d'autres types de messages, sous forme de données audio.

Selon le type d'application de téléservice, il arrive souvent que le vocabulaire des messages susceptibles d'être échangés ne soit pas illimité. Par exemple, dans un système interactif, il se peut que les seules réponses que l'on attende de l'utilisateur soient des instructions de type oui ou non. Dans ce cas, la transcription sémantique des messages vocaux peut être simplifiée : au lieu de transmettre les lettres O, U, I ou N, O, N selon la réponse de l'utilisateur, il suffit de transmettre un seul bit 1 ou 0. Le taux de compression obtenu par rapport à une transmission d'une réponse de type oui ou non sous forme de signal audio, est de l'ordre de plusieurs milliers au moins. De la même façon, pour chaque application impliquant des messages susceptibles d'être transmis fréquemment à travers le réseau de télécommunication 4, il est possible d'optimiser le codage en réduisant le nombre de bits utilisés pour coder ces messages récurrents. De manière idéale, lorsque le vocabulaire des messages échangés par le procédé de l'invention est parfaitement déterminé à l'avance, le codage sémantique des mots du vocabulaire peut être effectué de manière à minimiser le nombre de bits codant chaque mot, en tenant compte éventuellement de la probabilité de transmission de chaque mot (code à longueur variable). Cette transcription, qui est également de type sémantique, est généralement plus efficace que la transcription littérale de chaque lettre des mots désignant un sème dans une langue donnée. Le codage sémantique peut également être basé sur une prédiction des réponses de l'utilisateur et évoluer de manière dynamique au cours du dialogue (représentation de la prédiction sémantique). Comme mentionné, un minimum d'information caractérisant la voix et/ou la prosodie peut avantageusement être transmis par exemple sous la forme de marqueurs prosodiques incorporés dans la représentation sémantique.

Pour réduire encore la taille des messages, les messages sémantiques peuvent avantageusement être transcodés au moyen d'un code à réduction de redondance, par exemple au moyen d'un code de Huffman ou de Ziv-Lempel.

Les messages transmis codés sous forme sémantique contiennent l'information sur le contenu du message. Il est possible au niveau de chaque point d'accès réseau 2 d'attribuer une autre voix aux messages convertis en signaux audio. Par exemple, la voix allouée au même message pourra être masculine ou féminine selon le point d'accès 2 effectuant la conversion de texte en voix. Différents distributeurs d'information revendant les informations fournies par la plate-forme 1, peuvent de cette manière aisément personnaliser leur téléservice. Il est aussi possible d'instaurer au début de la communication un dialogue permettant à l'utilisateur de choisir une voix lui convenant.

La flexibilité du système peut encore être grandement étendue en adaptant la langue du contenu des signaux audio synthétisés à partir de la représentation sémantique des messages. Dans l'exemple ci-dessus, une réponse positive codée sous forme sémantique par un bit 1 peut avec la même facilité être convertie en un extrait de voix disant « oui », « yes » ou « ja » par exemple. Il en va de même de tous les messages échangés entre la plate-forme 1 et les points d'accès 2, lorsque leur sémantique peut être exprimée aisément en différentes langues. Chaque gestionnaire de point d'accès réseau 2 peut de cette manière convertir les messages codés sous forme sémantique en messages audio exprimés dans la langue prédominante à l'emplacement géographique considéré. Il est également possible d'offrir à l'utilisateur des possibilités de sélection de langue, par commandes vocales ou au moyen de son clavier, ou d'utiliser un algorithme qui reconnaisse automatiquement la langue de l'utilisateur appelant pour lui fournir ensuite les informations requises dans cette langue. La langue des messages audio synthétisés peut aussi être déterminée d'après les indications fournies par le fichier client des abonnés au service.

La description ci-dessus évoque plus particulièrement le cas d'un utilisateur appelant la plate-forme de téléservice 1 au moyen d'un combiné téléphonique fixe ou mobile. Il est cependant possible d'utiliser au lieu d'un téléphone n'importe quel type de second dispositif 30 permettant d'établir une communication à travers le réseau de télécommunication à transmission de paquets 4 et muni de moyens d'émission acoustiques, par exemple de haut-parleur, permettant d'écouter les messages vocaux diffusés par la plate-forme

1. Par exemple, le second dispositif 30 peut être constitué par un ordinateur personnel ou portable muni d'un modem et d'un haut-parleur (terminal intelligent).

Dans ce cas, certaines opérations déléguées dans l'exemple de mode de réalisation ci-dessus à la plate-forme 1 ou au point d'accès réseau 2 peuvent être réalisées par le terminal intelligent 30. Par exemple, si le terminal est muni de moyens permettant de le connecter directement au réseau à transmission de paquets 4, la synthèse de voix peut être effectuée par le terminal intelligent plutôt que par le point d'accès réseau 2. Le point d'accès réseau 2 peut même être supprimé si le terminal est en mesure de se connecter directement avec la plate-forme 1 à travers le réseau 4. Par exemple, dans le cas où le réseau 4 est un réseau de type internet, et la plate-forme 1 un serveur vocal branché sur internet, il est possible au moyen d'un ordinateur également connecté à internet de recevoir des messages vocaux codés sous forme sémantique et diffusés par la plate-forme 1 et de convertir ces messages codés en signaux audio, de préférence au moyen d'un logiciel de synthèse de voix de préférence téléchargé depuis la plate-forme 1. Le logiciel de synthèse de voix peut avantageusement être réalisé sous la forme d'un module informatique de type « plug-in » pour un logiciel de consultation (browser) de documents hypermédias transmis au travers du réseau internet.

L'invention peut en outre être appliquée à l'échange d'informations vocales entre deux utilisateurs, par exemple entre les utilisateurs des deux combinés téléphoniques 30 A et B sur la figure 1. La voix de l'utilisateur A est alors analysée et convertie en un message codé incluant une représentation sémantique par les moyens de traitement 20 du point d'accès réseau 2 le plus proche de A. Ce message codé est ensuite transmis à travers le réseau à transmission de paquets 4 vers le point d'accès réseau 2 le plus proche de B, où elle est synthétisée par les moyens de synthèse de l'unité de traitement 20 en un signal audio transmis à B via la connexion locale à travers le réseau d'accès 3. Les réponses de B sont transmises à A de manière similaire en sens inverse. Ce procédé permet par exemple des connexions longue distance à des tarifs imbattables, en raison de la compression très importante obtenue par le codage sémantique et de la transmission par paquets. La faible taille des

messages codés sous forme sémantique permet d'effectuer des dialogues en temps réel même lorsque le réseau 4 est très chargé, ce qui n'est pas toujours le cas des systèmes connus de téléphonie par internet (« internet vocal chat ») dans lesquels des données audionumériques sont transmises. Le prix à payer est naturellement une perte importante d'information relative à la diction ou à la prononciation de l'interlocuteur, dont la voix est remplacée par celle impersonnelle du synthétiseur de voix.

Pour pallier à cet inconvénient, il est possible si A et B ont fréquemment des conversations qu'ils s'expédient, une seule fois, un fichier contenant des paramètres caractérisant leur voix, et permettant aux moyens de synthèse de voix respectifs de synthétiser, à partir des messages reçus, des voix aussi proches que possible de celles de B, respectivement A.

Ce procédé peut en outre être utilisé en combinaison avec un procédé de téléphonie par internet classique, pour pallier au problème de lenteur de transmission de signaux audio à travers le réseau 4. Dans ce cas, les messages sont transmis de manière classique, sous forme de données audionumériques ou analogiques. Simultanément, un message contenant la transcription sémantique du message audio, de taille nettement inférieure à celle du message audio, est transmis en parallèle. Ce message est généralement reçu complètement bien avant le message audio. Si, après un intervalle de temps prédéterminé, le message audio n'est toujours pas parvenu à son destinataire, un nouveau message audio est synthétisé au niveau du destinataire à partir du message codé sous forme sémantique, et ce nouveau message est diffusé au destinataire sans attendre la réception complète du message audio original. La synthèse du nouveau message audio à partir du message codé sous forme sémantique peut avantageusement être effectuée en utilisant la connaissance des paramètres de la voix de l'interlocuteur, acquise lors d'échanges antérieurs de messages. Ce procédé permet en outre au destinataire d'avoir une transcription écrite du message vocal.

Ce procédé de communication direct entre deux interlocuteurs A et B s'avère particulièrement intéressant lorsque la langue des signaux audio synthétisés à partir des messages transmis codés sous forme sémantique est

différente pour A et pour B. Dans ce cas, il est possible de réaliser des systèmes de traduction automatique en temps réel sollicitant très peu le réseau de communication. Les messages vocaux de A, par exemple en français, sont convertis en messages ne contenant qu'une information sémantique au niveau du point d'accès réseau proche de A, puis transmis sous cette forme codée jusqu'au point d'accès réseau proche de B. Dans ce point d'accès réseau, un signal audio dans une autre langue, par exemple en japonais, est synthétisé à partir des informations sémantiques reçues, puis diffusé en japonais à l'attention de B. Les réponses de B sont transmises à A et traduites du japonais au français de manière opposée.

Revendications

1. Système de diffusion d'informations vocales à travers un réseau de télécommunication (4), comprenant les éléments suivants :

au moins un premier dispositif (1) connecté audit réseau de télécommunication et comprenant des moyens (10) pour diffuser des messages dans ledit réseau,

au moins un second dispositif (30) permettant d'établir à travers ledit réseau de télécommunication (4) une communication avec ledit premier dispositif (1), et muni de moyens d'émission acoustiques permettant d'écouter des signaux audio correspondant aux dits messages diffusés par ledit premier dispositif,

caractérisé en ce que lesdits messages sont transmis à travers ledit réseau de télécommunication sous forme codée, ledit code incluant une représentation sémantique des messages,

et en ce que lesdits seconds dispositifs (30) sont connectés audit réseau de télécommunication (4) par l'intermédiaire d'une ou plusieurs points d'accès au réseau (2) distribués et comprenant des moyens de synthèse de voix (20) aptes à convertir lesdits messages codés en signaux audio.

2. Système selon l'une des revendications précédentes, caractérisé en ce que ledit premier dispositif (1) est une plate-forme pour système de téléservice vocal automatique.

3. Système selon l'une des revendications précédentes, caractérisé en ce qu'il comporte plusieurs point d'accès au réseau distribués (2) géographiquement, et en ce que les messages expédiés par ledit premier dispositif (1) à destination d'un desdits seconds dispositifs (30) sont transmis sous forme codée jusqu'à un point d'accès au réseau (2) proche dudit second dispositif, synthétisés en signaux audio dans ledit point d'accès au réseau (2),

et transmis sous forme de signaux audio entre ledit point d'accès au réseau (2) et ledit second dispositif (30).

4. Système selon l'une des revendications précédentes, caractérisé en ce que ledit au moins un second dispositif (30) est relié audit au moins point d'accès au réseau (2) par l'intermédiaire d'un réseau d'accès (3).

5. Système selon l'une des revendications précédentes, caractérisé en ce que ledit réseau de télécommunication (4) est un réseau à transmission de paquets.

6. Système selon l'une des revendications précédentes, caractérisé en ce que ledit réseau de télécommunication (4) est un réseau de type internet, et en ce que ledit au moins un second dispositif (30) est un terminal internet.

7. Système selon la revendication précédente, caractérisé en ce que ledit réseau est un réseau à transmission de paquets, et en ce que plusieurs messages peuvent être multiplexés à l'intérieur de chaque paquet.

8. Système selon l'une des revendications précédentes, caractérisé en ce que lesdits points d'accès au réseau (2) comprennent des moyens de reconnaissance de parole (20) aptes à convertir des signaux audio provenant dudit au moins un second dispositif (30) en messages codés incluant une représentation sémantique desdits signaux, et des moyens pour expédier lesdits messages codés dans ledit réseau de télécommunication (4), et en ce que ledit au moins un premier dispositif (1) est apte à recevoir lesdits messages codés.

9. Système selon la revendication précédente, caractérisé en ce que ladite plate-forme pour système de téléservice vocal automatique (1) permet des dialogues interactifs et est apte à analyser lesdits messages codés expédiés par ledit au moins un second dispositif (30) et à diffuser une réponse dans ledit réseau de télécommunication (4) dépendant desdits messages reçus.

10. Système selon l'une des revendications précédentes, caractérisé en ce que lesdits moyens pour diffuser des messages dans ledit réseau comportent des moyens de stockage (10) desdits messages codés incluant une représentation sémantique et des moyens de lecture desdits messages stockés.

11. Système selon l'une des revendications précédentes, caractérisé en ce que lesdits moyens pour diffuser des messages dans ledit réseau comportent des moyens de stockage (10) desdits messages sous forme de signaux audio et des moyens de lecture desdits messages stockés et de reconnaissance de parole aptes à convertir lesdits signaux audio en messages codés incluant une représentation sémantique.

12. Système selon l'une des revendications précédentes, caractérisé en ce que ladite représentation sémantique optimise la longueur de messages codés susceptibles d'être transmis fréquemment à travers ledit réseau.

13. Système selon l'une des revendications précédentes, caractérisé en ce que ledit code inclut une représentation de la prédiction sémantique des messages.

14. Système selon l'une des revendications précédentes, caractérisé en ce que lesdits messages codés incluent des informations caractérisant la voix et/ou la prosodie.

15. Système selon l'une des revendications précédentes, caractérisé en ce que ledit au moins point d'accès au réseau (2) comprend une mémoire de type cache (22) mémorisant les messages expédiés par ledit premier dispositif (1).

16. Système selon la revendication précédente, caractérisé en ce que ladite mémoire de type cache (22) stocke les messages sous forme codée, incluant une représentation sémantique.

17. Système selon la revendication 15, caractérisé en ce que ladite mémoire de type cache (22) stocke les messages audio synthétisés à partir desdits messages codés.

18. Système selon l'une des revendications précédentes, caractérisé en ce qu'au moins certains moyens de synthèse de voix (20) aptes à convertir lesdits messages codés en signaux audio sont intégrés dans un second dispositif (30).

19. Système selon la revendication précédente, caractérisé en ce que lesdits moyens de synthèse de voix sont implémentés sous la forme d'un module « plug-in » pour un browser internet.

20. Système selon l'une des revendications précédentes, caractérisé en ce que lesdits moyens de synthèse de voix (21) aptes à convertir lesdits messages codés incluant une représentation sémantique en signaux audio sont programmés au moyen d'un module informatique, et en ce que ledit premier dispositif (1) et ledit au moins un point d'accès au réseau (2) comportent des moyens complémentaires et aptes à coopérer pour télécharger ledit module informatique depuis ledit premier dispositif (1) vers ledit au moins un point d'accès au réseau (2).

21. Système selon l'une des revendications 3 à 20, caractérisé en ce qu'il comporte plusieurs point d'accès au réseau distribués (2) sur plusieurs régions géographiques, et en ce que la langue desdits signaux audio synthétisés à partir desdits messages codés par lesdits moyens de synthèse de voix (20) dépend de la langue prédominante à l'emplacement géographique dudit point d'accès (2).

22. Système selon l'une des revendications précédentes, caractérisé en ce que la langue desdits signaux audio synthétisés à partir desdits messages codés est déterminée par l'utilisateur dudit au moins un second dispositif (30).

23. Plate-forme pour système de téléservice vocal automatique (1) apte à être connecté à un réseau de télécommunication (4) et comprenant des moyens pour générer des messages vocaux et les diffuser dans ledit réseau de télécommunication (4),

caractérisé en ce que lesdits messages sont diffusés dans ledit réseau de télécommunication sous forme codée, ledit code incluant une représentation sémantique desdits messages.

24. Plate-forme selon la revendication précédente, caractérisée en ce que lesdits moyens pour diffuser des messages dans ledit réseau sont constitués par des moyens de stockage (10) desdits messages codés et par des moyens de lecture desdits messages stockés.

25. Plate-forme selon la revendication 23, caractérisée en ce que lesdits moyens pour diffuser des messages dans ledit réseau comportent des moyens de stockage (10) desdits messages sous forme de signaux audio et des moyens de lecture desdits messages stockés et de reconnaissance de parole aptes à convertir lesdits signaux audio en messages codés incluant une représentation sémantique.

26. Plate-forme selon l'une des revendications 23 à 25, caractérisée en ce qu'elle permet des dialogues interactifs et est apte à analyser des messages codés incluant une représentation sémantique reçus à travers ledit réseau (4) et à diffuser une réponse dépendant desdits messages reçus.

27. Plate-forme selon l'une des revendications 23 à 26, caractérisée en ce que ladite représentation sémantique optimise la longueur de messages codés susceptibles d'être transmis fréquemment à travers ledit réseau.

28. Plate-forme selon l'une des revendications 23 à 27, caractérisé en ce que ledit code inclut une représentation de la prédiction sémantique des messages.

29. Plate-forme selon l'une des revendications 23 à 28, caractérisée en ce que lesdits messages codés incluent des informations caractérisant la voix et/ou la prosodie.

30. Plate-forme selon l'une des revendications 23 à 29, caractérisée en ce qu'elle est apte à être connectée à un réseau de télécommunication (4) à transmission de paquet.

31. Plate-forme selon l'une des revendications 23 à 30, caractérisée en ce qu'elle est apte à être connectée à un réseau de télécommunication (4) de type internet.

32. Plate-forme selon l'une des revendications 23 à 31, caractérisée par des moyens permettant de téléexporter à travers ledit réseau de télécommunication (4) un module informatique de synthèse de voix permettant de convertir lesdits messages codés incluant une représentation sémantique en signaux audio.

33. Point d'accès réseau (2) apte à être connecté à un réseau de télécommunication à transmission de paquets et comprenant des moyens pour établir une communication avec un autre dispositif (1) connecté audit réseau et pour recevoir des messages, notamment des messages vocaux diffusés par ledit autre dispositif,

caractérisé par des moyens de synthèse de voix (20) aptes à convertir des messages vocaux codés incluant une représentation sémantique diffusés par ledit autre dispositif (1) en signaux audio, et par des moyens pour expédier à travers un réseau d'accès (3) lesdits signaux audio vers au moins un second dispositif (30).

34. Point d'accès réseau (2) selon la revendication précédente, caractérisé par des moyens de reconnaissance de parole aptes à convertir des signaux audio reçus dudit au moins un second dispositif (30) en messages codés incluant une représentation sémantique, et par des moyens (21) pour

expédier lesdits messages codés sous forme sémantique dans ledit réseau de télécommunication à transmission de paquets (4).

35. Point d'accès réseau (2) selon l'une des revendications 33 ou 34, caractérisé en ce que ladite représentation sémantique optimise la longueur de messages codés susceptibles d'être transmis fréquemment à travers ledit réseau.

36. Point d'accès réseau (2) selon l'une des revendications 33 à 35, caractérisé en ce que ledit code inclut une représentation de la prédiction sémantique des messages.

37. Point d'accès réseau (2) selon l'une des revendications 33 à 36, caractérisé en ce que lesdits messages codés incluent des informations caractérisant la voix et/ou la prosodie.

38. Point d'accès réseau (2) selon l'une des revendications 33 à 37, caractérisé par une mémoire de type cache (22) pour les messages reçus.

39. Point d'accès réseau (2) selon la revendication précédente, caractérisé en ce que ladite mémoire de type cache (22) stocke les messages sous forme codée incluant une représentation sémantique.

40. Point d'accès réseau (2) selon la revendication précédente, caractérisé en ce que ladite mémoire de type cache (22) stocke les messages audio synthétisés à partir desdits messages codés.

41. Point d'accès réseau (2) selon l'une des revendications 33 à 40, caractérisé par des moyens pour expédier des signaux audio synthétisés à partir desdits messages codés ou pour recevoir des signaux audio à coder sur ledit réseau de télécommunication.

42. Point d'accès réseau (2) selon l'une des revendications 33 à 41, caractérisé en ce que ledit réseau de télécommunication (4) est un réseau de type internet.

43. Point d'accès réseau (2) selon l'une des revendications 33 à 42, caractérisé par des moyens permettant de choisir la langue du contenu desdits signaux audio obtenus suite à ladite conversion des messages codés sous forme sémantique.

44. Procédé de communication entre une plate-forme pour système de téléservice vocal automatique (1), et au moins un second dispositif (30) connecté à ladite plate-forme par l'intermédiaire d'un point d'accès réseau (2) et d'un réseau de télécommunication , caractérisé par les étapes suivantes :

ledit second dispositif (30) requiert l'établissement d'une connexion à travers ledit point d'accès réseau (2) avec ladite plate-forme (1),

ladite connexion est établie à travers ledit réseau de télécommunication (4),

ladite plate-forme (1) diffuse au moins un message vocal codé sous forme sémantique transmis à travers ledit réseau de télécommunication (4) vers ledit point d'accès réseau (2),

ledit point d'accès réseau (2) convertit grâce à des moyens de synthèse de voix (20) lesdits messages codés sous forme sémantique en signaux audio,

lesdits signaux audio sont transférés vers ledit second dispositif (30) où ils peuvent être écoutés par l'utilisateur dudit second dispositif.

45. Procédé selon la revendication précédente, caractérisé en ce qu'il comporte en outre au moins une fois la succession d'opérations suivantes :

ledit second dispositif (30) transmet au moins un signal audio audit point d'accès réseau (2),

ledit point d'accès réseau (2) convertit le ou lesdits signaux audio en message codé sous forme sémantique grâce à des moyens de reconnaissance de parole (20),

ledit point d'accès réseau (2) expédie vers ladite plate-forme (1), à travers ledit réseau de télécommunication (4), le ou lesdits messages codés sous forme sémantique,

ladite plate-forme (1) reçoit le ou lesdits messages codés sous forme sémantique.

46. Procédé selon la revendication précédente, caractérisé par une étape au cours de laquelle ladite plate-forme (1) analyse lesdits messages vocaux codés sous forme sémantique expédiés par ledit au moins un second dispositif (30) et détermine une réponse à expédier dans ledit réseau de télécommunication (4) dépendant desdits messages reçus.

47. Procédé selon l'une des revendications 44 à 46, caractérisé en ce que ledit codage de message sous forme sémantique optimise la longueur de messages codés susceptibles d'être transmis fréquemment à travers ledit réseau.

48. Procédé selon l'une des revendications 44 à 47, caractérisé en ce que ledit code inclut une représentation de la prédiction sémantique des messages.

49. Procédé selon l'une des revendications 44 à 48, caractérisé en ce que ledit codage de message sous forme sémantique utilise des informations caractérisant la voix et/ou la prosodie .

50. Procédé selon l'une des revendications 44 à 49, caractérisé en ce que les messages reçus de ladite plate-forme (1) sont stockés dans une mémoire de type cache (22).

51. Procédé selon la revendication précédente, caractérisé en ce que ladite mémoire de type cache (22) stocke les messages codés sous forme sémantique.

52. Procédé selon la revendication 50, caractérisé en ce que ladite mémoire de type cache (22) stocke les messages décodés sous forme de signaux audio.

53. Procédé selon l'une des revendications 44 à 52, caractérisé en ce que ledit réseau de télécommunication (4) est un réseau à transmission de paquet.

54. Procédé selon l'une des revendications 44 à 53, caractérisé en ce que ledit réseau de télécommunication (4) est un réseau de type internet.

55. Procédé selon la revendication précédente, caractérisé en ce que ledit réseau est un réseau à transmission de paquets, et en ce que plusieurs messages peuvent être multiplexés à l'intérieur de chaque paquet.

56. Support de données informatiques pouvant être lu par un dispositif programmable connecté à un réseau internet, stockant un programme d'instructions exécutable par ledit dispositif pour exécuter les opérations effectuées par ledit point d'accès réseau dans le procédé de l'une des revendications 44 à 55.

57. Support de données informatiques pouvant être lu par un dispositif programmable connecté à un réseau internet, stockant un programme d'instructions exécutable par ledit dispositif pour exécuter les opérations effectuées par ladite plate-forme pour système de téléservice vocal automatique (1) dans le procédé de l'une des revendications 55 à 55.

58. Système de diffusion d'informations vocales à travers un réseau de télécommunication à transmission de paquets (4), comprenant les éléments suivants :

des moyens (10) de diffusion de messages dans ledit réseau et de gestion de dialogue avec l'utilisateur,

des moyens (20) de synthèse de voix,

caractérisé en ce que lesdits moyens de diffusion de messages dans ledit réseau et de gestion de dialogue avec l'utilisateur sont centralisés dans une plate-forme pour système de téléservice vocal automatique (1), en ce que lesdits moyens de synthèse de voix sont distribués dans une pluralité de points d'accès réseau (2) répartis dans ledit réseau (4), et en ce que lesdits messages sont transmis entre ladite plate-forme (1) et lesdits points d'accès réseau (2) sous forme codée, ledit code incluant une représentation sémantique des messages.

59. Système selon la revendication précédente, caractérisé en ce que l'utilisateur peut écouter lesdits messages en établissant une connexion avec un desdits points d'accès réseau à travers un réseau de télécommunication (3).

1/1

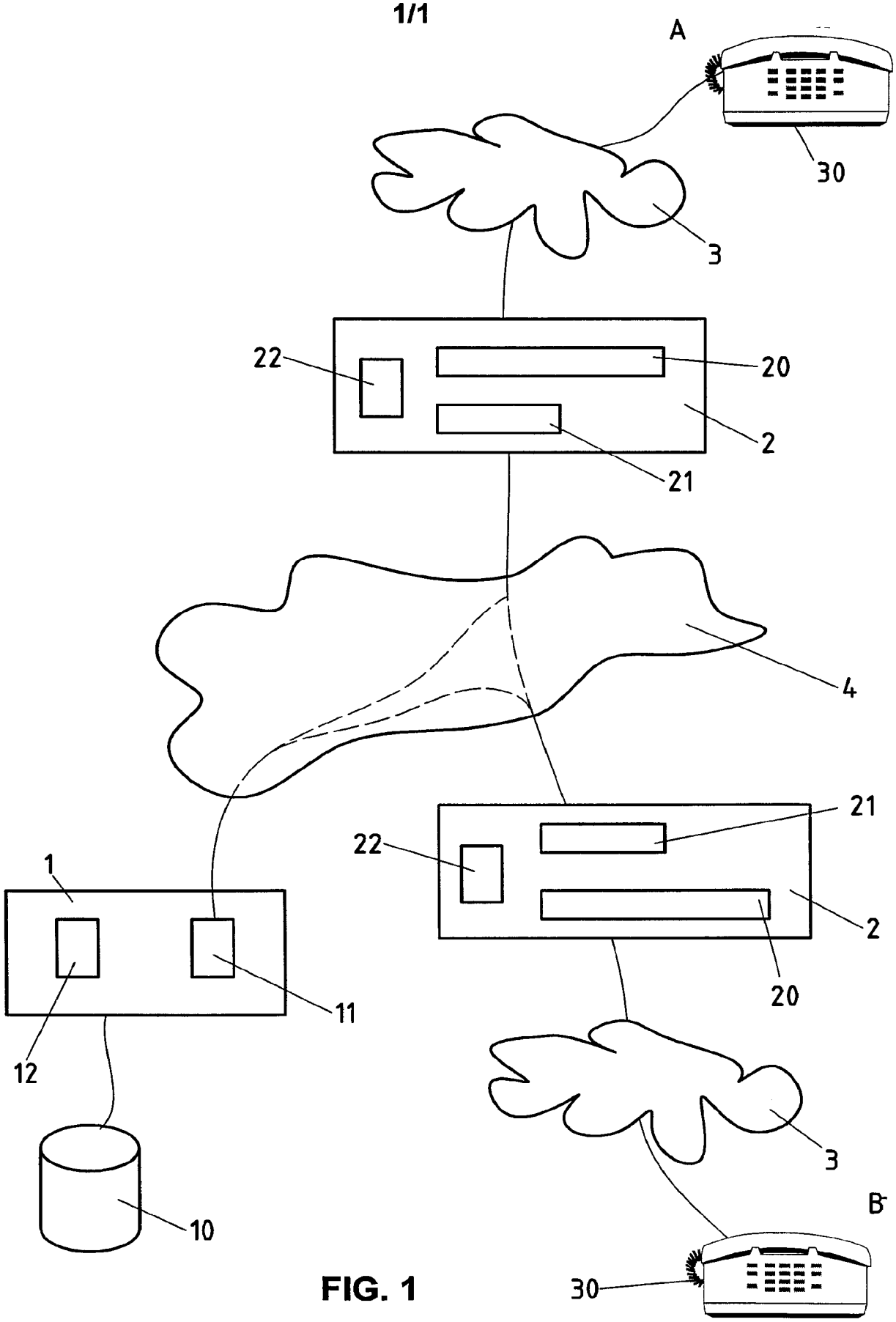


FIG. 1

INTERNATIONAL SEARCH REPORT

Int. ,onal Application No
PCT/CH 97/00246

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04M3/50

According to International Patent Classification(IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04M

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 96 18258 A (VOICE CONTROL SYSTEMS INC) 13 June 1996 see page 1, line 1 - page 3, line 11; claims 1,7,10,30; figure 1 ---	1,2,23, 33,44,58
A	GB 2 165 969 A (BRITISH TELECOMM) 23 April 1986 see page 1, line 1 - line 54; claims 1,16; figures 1,2 ---	1,2,23, 33,44,58
A	US 5 150 410 A (BERTRAND JOHN) 22 September 1992 see column 1, line 1 - column 2, line 20; claim 1; figure 5 --- -/--	1,33

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

2

Date of the actual completion of the international search	Date of mailing of the international search report
9 March 1998	17/03/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	De Haan, A.J.

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Int tional Application No
PCT/CH 97/00246

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	GB 2 249 701 A (BRITISH TELECOMM) 13 May 1992 see page 1, line 1 - line 31; claim 1; figure 1 -----	1

2

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Int. ional Application No
PCT/CH 97/00246

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9618258 A	13-06-96	AU 4739596 A	26-06-96
		CA 2206870 A	13-06-96
		EP 0795246 A	17-09-97
GB 2165969 A	23-04-86	NONE	
US 5150410 A	22-09-92	NONE	
GB 2249701 A	13-05-92	NONE	

Form PCT/ISA/210 (patent family annex) (July 1992)

RAPPORT DE RECHERCHE INTERNATIONALE

De[illegible]ie Internationale No
PCT/CH 97/00246

A. CLASSEMENT DE L'OBJET DE LA DEMANDE
CIB 6 H04M3/50

Selon la classification internationale des brevets (CIB) ou à la fois selon la classification nationale et la CIB

B. DOMAINES SUR LESQUELS LA RECHERCHE A PORTE

Documentation minimale consultée (système de classification suivi des symboles de classement)
CIB 6 H04M

Documentation consultée autre que la documentation minimale dans la mesure où ces documents relèvent des domaines sur lesquels a porté la recherche

Base de données électronique consultée au cours de la recherche internationale (nom de la base de données, et si cela est réalisable, termes de recherche utilisés)

C. DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie °	Identification des documents cités, avec, le cas échéant, l'indication des passages pertinents	no. des revendications visées
A	WO 96 18258 A (VOICE CONTROL SYSTEMS INC) 13 juin 1996 voir page 1, ligne 1 - page 3, ligne 11; revendications 1,7,10,30; figure 1 ---	1,2,23, 33,44,58
A	GB 2 165 969 A (BRITISH TELECOMM) 23 avril 1986 voir page 1, ligne 1 - ligne 54; revendications 1,16; figures 1,2 ---	1,2,23, 33,44,58
A	US 5 150 410 A (BERTRAND JOHN) 22 septembre 1992 voir colonne 1, ligne 1 - colonne 2, ligne 20; revendication 1; figure 5 ---	1,33

-/--

[X] Voir la suite du cadre C pour la fin de la liste des documents

[X] Les documents de familles de brevets sont indiqués en annexe

° Catégories spéciales de documents cités:

"A" document définissant l'état général de la technique, non considéré comme particulièrement pertinent

"E" document antérieur, mais publié à la date de dépôt international ou après cette date

"L" document pouvant jeter un doute sur une revendication de priorité ou cité pour déterminer la date de publication d'une autre citation ou pour une raison spéciale (telle qu'indiquée)

"O" document se référant à une divulgation orale, à un usage, à une exposition ou tous autres moyens

"P" document publié avant la date de dépôt international, mais postérieurement à la date de priorité revendiquée

"T" document ultérieur publié après la date de dépôt international ou la date de priorité et n'appartenant pas à l'état de la technique pertinent, mais cité pour comprendre le principe ou la théorie constituant la base de l'invention

"X" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme nouvelle ou comme impliquant une activité inventive par rapport au document considéré isolément

"Y" document particulièrement pertinent; l'invention revendiquée ne peut être considérée comme impliquant une activité inventive lorsque le document est associé à un ou plusieurs autres documents de même nature, cette combinaison étant évidente pour une personne du métier

"&" document qui fait partie de la même famille de brevets

2

Date à laquelle la recherche internationale a été effectivement achevée	Date d'expédition du présent rapport de recherche internationale
9 mars 1998	17/03/1998

Nom et adresse postale de l'administration chargée de la recherche internationale	Fonctionnaire autorisé
Office Européen des Brevets, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	De Haan, A.J.

Formulaire PCT/ISA/210 (deuxième feuille) (juillet 1992)

RAPPORT DE RECHERCHE INTERNATIONALE

Der e internationale No
PCT/CH 97/00246

C.(suite) DOCUMENTS CONSIDERES COMME PERTINENTS

Catégorie	Identification des documents cités. avec,le cas échéant, l'indicationdes passages pertinents	no. des revendications visées
A	GB 2 249 701 A (BRITISH TELECOMM) 13 mai 1992 voir page 1, ligne 1 - ligne 31; revendication 1; figure 1	1

2

Formulaire PCT/ISA/210 (suite de la deuxième feuille) (juillet 1992)

RAPPORT DE RECHERCHE INTERNATIONALE

Renseignements relatifs aux membres de familles de brevets

De: le internationale No
PCT/CH 97/00246

Document brevet cité au rapport de recherche	Date de publication	Membre(s) de la famille de brevet(s)	Date de publication
WO 9618258 A	13-06-96	AU 4739596 A	26-06-96
		CA 2206870 A	13-06-96
		EP 0795246 A	17-09-97
GB 2165969 A	23-04-86	AUCUN	
US 5150410 A	22-09-92	AUCUN	
GB 2249701 A	13-05-92	AUCUN	

Formulaire PCT/ISA/210 (annexe familles de brevets) (juillet 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30 — A1

(11) International Publication Number: WO 99/17227

(43) International Publication Date: 8 April 1999 (08.04.99)

(21) International Application Number: PCT/GB98/02920

(22) International Filing Date: 28 September 1998 (28.09.98)

(30) Priority Data:
08/939,277 29 September 1997 (29.09.97) US

(71) Applicant: INTERNATIONAL BUSINESS MACHINES CORPORATION [US/US]; New Orchard Road, Armonk, NY 10504 (US).

(71) Applicant *(for MC only)*: IBM UNITED KINGDOM LIMITED [GB/GB]; North Harbour, P.O. Box 41, Portsmouth, P.O. Box 41, Hampshire P06 3AU (GB).

(72) Inventors: MALKIN, Peter, Kenneth; 64 Bramble Brook Road, Ardsley, NY 10502 (US). YU, Philip, Shi-lung; 18 Stonowaye, Chappaqua, NY 10514 (US).

(74) Agent: DAVIES, Simon, Robert; IBM United Kingdom Limited, Intellectual Property Dept., Hursley Park, Winchester, Hampshire SO21 2JN (GB).

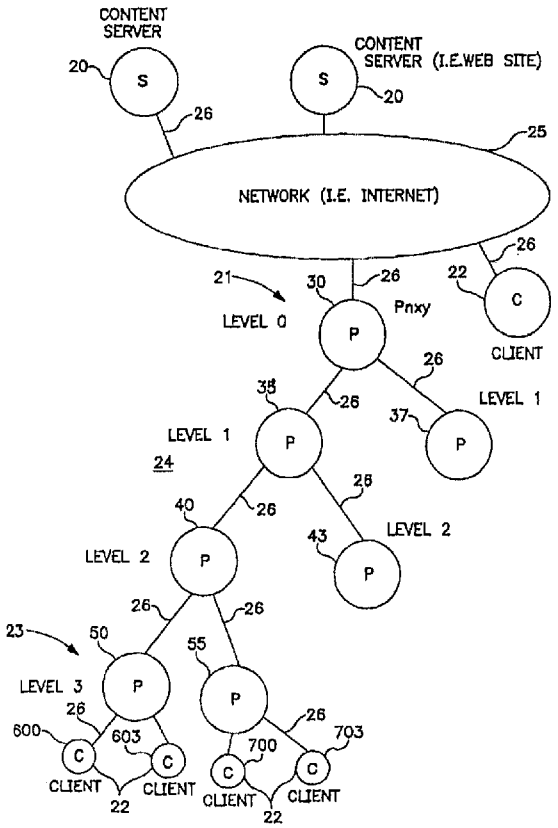
(81) Designated States: CA, CN, CZ, HU, IL, JP, KR, PL, RU, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: METHOD AND SYSTEM FOR PREFETCHING INFORMATION

(57) Abstract

A method and system are disclosed for prefetching data for clients associated with a proxy server hierarchy and an enhanced content server data transfer to the clients. The method for prefetching data identifies data access patterns and prefetches select information based on a dynamic interpretation of the data access patterns. The content server or proxy server identifies data access reference patterns of clients associated with the content server or the proxy server hierarchy. The data access patterns are preferably stored in a statistics table. The content server generates prefetch hint information based on inputs from the statistics table. The prefetch hint information annotates requested information or objects requested by the clients. The prefetch hint information optimally is updated as the requested information passes through the proxy server hierarchy. The prefetch hint information annotates the requested information requested by the clients. The ultimate decision to prefetch select information for the clients is made based on the prefetch hint information and prefetch hint values. The prefetch hint value is calculated based on local node-specific factors, such as storage constraints and access time.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	ES	Spain	LS	Lesotho	SI	Slovenia
AM	Armenia	FI	Finland	LT	Lithuania	SK	Slovakia
AT	Austria	FR	France	LU	Luxembourg	SN	Senegal
AU	Australia	GA	Gabon	LV	Latvia	SZ	Swaziland
AZ	Azerbaijan	GB	United Kingdom	MC	Monaco	TD	Chad
BA	Bosnia and Herzegovina	GE	Georgia	MD	Republic of Moldova	TG	Togo
BB	Barbados	GH	Ghana	MG	Madagascar	TJ	Tajikistan
BE	Belgium	GN	Guinea	MK	The former Yugoslav Republic of Macedonia	TM	Turkmenistan
BF	Burkina Faso	GR	Greece			TR	Turkey
BG	Bulgaria	HU	Hungary	ML	Mali	TT	Trinidad and Tobago
BJ	Benin	IE	Ireland	MN	Mongolia	UA	Ukraine
BR	Brazil	IL	Israel	MR	Mauritania	UG	Uganda
BY	Belarus	IS	Iceland	MW	Malawi	US	United States of America
CA	Canada	IT	Italy	MX	Mexico	UZ	Uzbekistan
CF	Central African Republic	JP	Japan	NE	Niger	VN	Viet Nam
CG	Congo	KE	Kenya	NL	Netherlands	YU	Yugoslavia
CH	Switzerland	KG	Kyrgyzstan	NO	Norway	ZW	Zimbabwe
CI	Côte d'Ivoire	KP	Democratic People's Republic of Korea	NZ	New Zealand		
CM	Cameroon			PL	Poland		
CN	China	KR	Republic of Korea	PT	Portugal		
CU	Cuba	KZ	Kazakstan	RO	Romania		
CZ	Czech Republic	LC	Saint Lucia	RU	Russian Federation		
DE	Germany	LI	Liechtenstein	SD	Sudan		
DK	Denmark	LK	Sri Lanka	SE	Sweden		
EE	Estonia	LR	Liberia	SG	Singapore		

METHOD AND SYSTEM FOR PREFETCHING INFORMATION

The present invention is related to a method and apparatus for the dynamic prefetching of information or objects for clients, wherein the clients are preferably associated with hierarchy of proxy servers, such as may be involved in prefetching Web objects on the World Wide Web.

GLOSSARY OF TERMS

Internet: The network of networks and gateways that use the TCP/IP suite of protocols.

Client: A client is a computer which issues commands to the server which performs the task associated with the command.

Server: Any computer that performs a task at the command of another computer is a server. A Web server typically supports one or more clients.

World Wide Web (WWW or Web): The Internet application that lets users seeking information on the Internet switch connection from server to server and database to database by choosing ("clicking on") highlighted words or phrases of interest (known as hyperlinks). An Internet WWW server supports clients and provides information to the clients. The Web, which can be considered as the Internet with all of the resources addressed as URLs, uses HTML to display the information corresponding to URLs, and provides a point-and-click interface to other URLs.

Universal Resource Locator (URL): The URL is the method to uniquely identify or address information on the Internet, and may be considered a Web document version of an e-mail address. URLs may be accessed with a Hyperlink. An example of a URL identification is "http://www.philipyu.com:80/table.html". The URL has four components. Starting from the left of the example, the first component specifies the protocol to use (in this case http), separated from the rest of the locator by a ":". The next component is the hostname or IP address of the target host; this component is delimited by the double slash "//" on the left and on the right by a single slash "/" (or optionally a ":"). The port number is an optional component, and is delimited on the left from the hostname by a ":" and on the right by a single slash "/". The fourth component is the actual file name or program name (in this example, table.html), using further single slashes to separate directories and subdirectories in conventional fashion. In this example, the ".html" extension means that this is an HTML file.

HyperText Markup Language (HTML): HTML is a language used by Web servers to create and connect documents that are viewed by Web clients. HTML generates Hypertext documents.

Hypertext transfer protocol (HTTP): HTTP is an example of a stateless protocol, in which every request from a client to a server is treated independently. The server has no record of previous connections.

At the beginning of a URL, "http:" indicates the requesting client and target server should communicate using the HTTP protocol regarding the specified resource.

Internet Browser or Web browser: A graphical interface tool that runs Internet protocols such as HTTP, and display results on the user's screen. The browser can act as an Internet tour guide, complete with pictorial desktops, directories and search tools for when a user "surfs" the Internet. In this application the Web browser is a client service which communicates with the World Wide Web.

Client cache: Client caches are typically used as a primary group (caches) of objects accessed by the client. In the WWW environment, client caches are typically implemented by web browsers and may cache objects accessed during a current invocation, i.e., a non persistent cache, or may cache objects across invocations.

Caching proxies: Specialized servers in a network which act as agents on the behalf of the client to locate an object, possibly returning a cached copy. Caching proxies typically serve as secondary or higher level caches, because they are invoked as a result of cache misses from client caches.

META-tagging: The association of information with a given object. For example, in HTTP, information can be associated with both requests and responses in the fields of the HTTP header. For example, an HTTP client can specify the URL from which a returned page was requested.

***

As the World Wide Web (WWW or Web) increases in popularity, traffic over the Internet has increased considerably. The Web has now become one of the primary bottlenecks on network performance. For example, a Web user may experience noticeable delays when documents or information are requested by the user who is connected to a server via a slow network link. The Web's transferring of the information over the network to the user further increases the level of traffic over the network. The increased traffic reduces the bandwidth available for the requests of other users, contributing to additional potential delays.

To reduce access latencies for clients, the prior art has stored or cached copies of popular documents of information closer to the user at network nodes, from which the access latencies are more acceptable. The caching can be implemented at various points on the network. For example, a large university or corporation may have its own local cache, from which all the users subscribing to that network may fetch documents. A local cache may be implemented in the form of a specialized server.

A specialized server, which is called a caching proxy, may act as an agent on the behalf of clients to locate any potential cached copy of

requested information. Caching proxies usually serve as secondary or higher level caches because caching proxies are concerned only with misses left over from failed client caches. Client caches are built into various Web browsers. Client caches may either store only the document accesses during the current invocation (nonpersistent cache such as Mosaic) or may cache documents across invocations.

Previous work on the caching of Web documents has mainly been in the area of on-demand caching. On-demand caching means a caching decision is made after an object is requested. For example, Abrams et. al., "Caching Proxies: Limitations and Potentials", Proc. 4th International World Wide Web Conference, 1996, describes different on-demand caching policies that have been examined for the Web.

In an Internet Service Provider (ISP) network or a corporate network, different regional proxies or departmental proxies in the proxy hierarchy may observe different reference behavior or face different network traffic and delay. The presence of the proxy hierarchy further complicates the prefetch decision on whether all the intermediate proxies, the client, or a subset of them should perform the prefetch. Thus, a need exists for a prefetching method and system that can effectively coordinate prefetching activities with respect to the proxy server hierarchy and associated clients.

Accordingly, the present invention provides a method for prefetching select information from a content server to make the select information readily accessible for one or more clients associated with the content server via a communications network and one or more proxy servers, the method comprising the steps of:

requesting information from at least one of the proxy servers;

establishing prefetch hint information for said clients based on past retrievals or accesses of the requested information by any of said clients;

annotating the requested information with said prefetch hint information; and

prefetching select information from any of said servers based upon the prefetch hint information and the requested information.

Typically, the requested information comprises requested objects, and the select information comprises prefetch candidate objects, such as Internet data. In this case the requested object can reference the select object through a hypertext transfer protocol (HTTP) link associated with the Internet data, the proxy server identifying the relationship of the select object to the requested object through the HTTP protocol of the

requested object universal resource locator, or URL, stored in an object header of the requested object.

Preferably the step of establishing prefetch hint information includes the steps of: tracking data reference patterns associated with said clients; organizing the data reference patterns into groups of object identifiers, the groups of object identifiers organized according to relationships between the requested objects and prefetch candidate objects; and determining which of said groups have sufficient relationships between the prefetch candidate objects and the requested objects to warrant prefetching of said prefetch candidate objects as the select information. Thus the data reference pattern of said clients is advantageously based upon the past retrievals or prior accesses of the requested information, the data reference pattern being selected to represent any common trends in past retrievals or accesses by different ones of said clients associated with one of said proxy servers. This can then be used to define the prefetch hint information.

The steps of tracking and organising preferably include establishing databases for storing reference data patterns in the proxy servers; arranging fields within each of said databases, the fields including object identifiers and object counts, the object identifiers distinguishing the requested objects from one another; the object counts indicating the number of times the requested objects were requested as observed by one of said servers; and associating access lists with said databases, each of said access lists containing one or more prefetch candidate objects associated with each requested object.

Such a database can be used to define the relationship between requested objects and prefetch candidate objects and to track access statistics on the requested objects and the prefetch candidate objects; the access statistics including a parent-object access frequency of requested objects and a child-object access frequency of the prefetch candidate objects; the parent-object access frequency representing the number of times in which a particular one of the requested objects is requested by at least one of said clients; the child-object access frequency representing the number of times in which one or more of said prefetch candidate objects are actually requested proximately following the accessing of the particular one. This allows the prefetch list of prefetch candidate objects to be selected based upon the ratio of the child-object access frequency to the parent-object access frequency associated with the child-object access frequency, the prefetch list containing only prefetch candidate objects with ratios that exceed a threshold probability.

Generally only a subset of prefetch candidate objects will actually be prefetched and received at one of said proxy servers or one of said clients, typically the selection of this subset of the prefetch candidate objects as select information being based on factors such as the available storage size at one of said proxy servers and/or the access time at one of said proxy servers. Thus the prefetch candidate objects are prefetched in accordance with the priority associated with some corresponding prefetch value.

In the preferred embodiment, the step of annotating the requested information comprises the step of placing the prefetch hint information in a prefetch label having a Platform for Internet Content Selection (PICS) format, or a label protocol compatible with the PICS format. In a particular implementation of this (PICS) format, an object identifier category and a referrer identifier category are defined, and said method further includes the step of returning the referrer identifier category to the content server to identify any prefetch candidate objects which were actually referenced. Organizing the prefetch hint information according to a Platform for Internet Content Selection protocol, or a PICS-compatible protocol utilises a format already supported by many browswers.

Typically there is a hierarchy of said proxy servers. Defining a sending node and a receiving node in this hierarchy, the prefetch hint information is preferably modified from an initial state to a final state, the initial state determined by the sending node, the final state determined by the receiving node. It is the prefetch hint information in said final state which then forms the basis for the ultimate prefetch decision (assuming that prefetching select information is initiated by the receiving node).

In other words, the prefetch hint information can be interpreted and updated as it is transmitted through the hierarchy from the proxy servers to the clients. Preferably a proxy server updates the prefetch hint information if local references to the requested information exist and if the count of the local-level server exceeds a minimum threshold count (i.e. reliable statistics are therefore available locally). The prefetch hint information can also be updated based on caching status in the proxy hierarchy (eg to indicate where in the hierarchy a particular prefetch candidate object may be cached).

The invention further provides a data processing system for prefetching select information from a content server over a network, said system including one or more proxy servers associated with the network, and clients for communicating with the content server and one or more of the proxy servers over the network; said system comprising:

means for requesting information from at least one of the proxy servers;

means for establishing prefetch hint information for said clients based on past retrievals or accesses of the requested information by any of said clients;

means for annotating the requested information with said prefetch hint information; and

means for prefetching select information from any of said servers based upon the prefetch hint information and the requested information.

In a preferred embodiment, the requested information comprises requested objects, and the select information comprises prefetch candidate objects, wherein each of said proxy servers and at least one of the clients has a memory having at least a logical separation between a main cache and a prefetch cache, wherein the prefetched select information is stored in the prefetch cache.

It is preferred that the prefetching means prefetches objects and stores the prefetched objects in the prefetch cache based on a priority scheme, the priority scheme storing objects with a highest prefetch value first, with regard to the available memory of the prefetch cache, the prefetch value being based upon an individual or aggregate reference probability to the prefetch objects by one or more clients. In particular, the prefetch value may be based on an estimated access probability that a prefetch candidate object on the prefetch hint information list will be accessed by particular ones of the clients and on an access cost of the object based on requisite storage size for storing select ones of the prefetch candidate objects.

In a preferred embodiment, the prefetching means comprises a prefetch handler including computing means for computing said prefetch value, and evaluating means for evaluating the prefetch value computed by the computing means; said computing means computing the prefetch value for each prefetch object identifier on the prefetch hint information list; said evaluating means evaluating the prefetch values of multiple candidate prefetch objects and prioritizing the caching of candidate prefetch objects associated with the highest prefetch values on the prefetch hint information list. The prefetching means may comprise a first prefetch handler in at least one of said proxy servers and a second prefetch handler in at least one of said clients (i.e. prefetching may be performed at different/multiple locations in the hierarchy).

A preferred embodiment also includes a client object request handler stored in the memory of one of said clients, the client object request handler searching the main cache and prefetch cache for a requested object

prior to requesting the requested object from a proxy server or the network, the client object request handler notifying the content server or proxy server if the object is located in the prefetch cache by generating a prefetch referenced message to the server indicative that the prefetch object was actually accessed by the client. Similarly it is preferred that at least one of said proxy servers includes a prefetch referenced message handler for forwarding a prefetch referenced message after invoking a cache manager routine if a requested object is located in the prefetch cache. This ensures that the server can maintain an accurate record of access statistics even if it is not directly contacted for a child object because it is available from a prefetch cache at the client or in a proxy server.

The invention further provides a content or proxy server for use in such a data processing system, wherein the requested information comprises requested objects, and wherein the select information comprises prefetch candidate objects, said server including:

a statistics table storing parent object identifiers and associated child object identifiers, the statistics table containing first counts of parent objects, the first counts representing the frequency with which parent objects are referenced by the clients, the statistics table containing second counts of child objects, the second counts representing the frequency with which the corresponding child objects are referenced after select ones of said parent objects are referenced; and

generating means for generating the prefetch hint information list based on information in the statistics table.

In one embodiment, the generating means accepts inputs of first counts, second counts, child object identifiers, and parent object identifiers from the statistics table, the generating means including evaluating means for evaluating the access relationship between parent objects and their associated child objects, the evaluating means placing child object identifiers on a prefetch hint information list if the child objects are referenced with adequate frequencies proximately and subsequently prior to references by the clients to corresponding parent objects. The adequate frequencies can be based on the ratio of child accesses to corresponding parent accesses. Typically this involves calculating a ratio of the second count to the first count for each of said parent object identifiers; comparing the ratio to a hint threshold; and grouping or associating child objects and associated parent objects on the prefetch hint information list if the ratio exceeds the hint threshold.

Preferably the server further comprises statistics updating means for updating the statistics table based upon a child object actually

requested; the statistics updating means first searching for an identity of a referring one of said parent objects, the referring one, or referrer object, being referenced prior to the child object actually requested, the child object actually requested having its corresponding second count updated in the statistics table. The statistics table should also be updated upon receipt of a prefetch referenced message from a prefetch referenced message handler.

It is preferred that where the server is a proxy server, it further comprises prefetch updating means for receiving and updating the prefetch hint information list. The prefetch updating means compares the first counts to first local threshold counts, and invokes the generating means for each of the child object identifiers for which the first counts exceeds the first local threshold counts, the prefetch updating means keeping the original prefetch hint information list if the first counts do not exceed the first local threshold counts. In a more sophisticated approach, the prefetch updating means compares the first counts to first and second local threshold counts, and (i) keeps the original prefetch hint information list if the first counts do not exceed the first local threshold counts; (ii) modifies the original prefetch hint information list if the first counts exceed the first local threshold counts but not the second local threshold counts; and (iii) replaces the original prefetch hint information list if the first counts exceed both the first and second local threshold counts.

Preferably the server further includes: a proxy object handler for returning a requested object to a lower level proxy server or client that requested the requested object, the proxy object hander returning the requested object and updating the statistics table accordingly if the requested object is not a prefetch object; and a prefetch cache manager for managing a prefetch cache. Thus the proxy object handler returns a requested object from the prefetch cache to a requesting client or proxy server, and marks the object as deletable from the prefetch cache if the object was requested pursuant to a prefetch request. The proxy object handler sends an object request to another server if the requested object is not in a main cache or the prefetch cache. The proxy object handler updates the statistics table accordingly if the requested object is not associated with a prefetch request and if the requested object is located in the prefetch cache or the main cache.

Viewed from another aspect, the invention provides a method for prefetching select objects from a content server to make the select

objects readily available for clients associated with one or more proxy servers via a communications network; the method comprising the steps of:

a) requesting requested objects from at least one of said proxy servers;

b) identifying data reference patterns of corresponding clients based on the past retrievals or accesses of requested objects by the clients;

c) developing prefetch hint information into a prefetch hint information list from the data reference patterns;

d) annotating the requested objects with the prefetch hint information;

e) prefetching select objects from at least one of said servers based upon the prefetch hint information and currently requested objects.

Thus the select information which is prefetched comprises a prefetch candidate object; and the step establishing prefetch hint information further comprises establishing prefetch hint information at a sending node in a hierarchy of said proxy servers; where the prefetch candidate object is passed through a hierarchy of said proxy servers; and the select information to prefetch at a receiving node in the hierarchy is based on the prefetch hint information.

There is typically a proxy server hierarchy associated with the network, the proxy server hierarchy including a lower-level server more closely associated with the clients than other proxy servers. In one preferred embodiment, the prefetch hint information is updated through the proxy server hierarchy if local references to the requested object exist and if the count of the local-level server exceeds a minimum threshold count; alternatively (or additionally) the prefetch hint information may be updated by analyzing the caching status at higher level hierarchy, the highest level of the hierarchy being most closely associated with the network.

Viewed from another aspect, the invention provides a processing system for enhancing a flow of data from servers to clients along a network, the processing system comprising:

proxy servers associated with the network;

a content server associated with the network;

clients configured for communicating with one or more of said proxy servers;

a statistics table being stored in at least one of said proxy servers, the statistics table storing parent object identifiers and associated child object identifiers, the statistics table containing first counts of parent objects, the first counts representing the frequency in which parent objects are referenced by the clients, the statistics table

containing second counts of child objects, the second counts representing the frequency in which the corresponding child objects are referenced after select ones of said parent objects are referenced; and

generating means for generating a prefetch hint information list by accepting inputs of the first counts and the corresponding second counts to derive the prefetch hint information list, said generating means including:

calculating means for calculating a ratio of the second count to the first count for each of said parent object identifiers;

comparing means for comparing the ratio to a hint threshold, the comparing means grouping or associating child objects and associated parent objects on the prefetch hint information list if the ratio exceeded the hint threshold; and

prefetching means for prefetching child objects based on the prefetch hint information and parent objects actually requested by the clients.

In a preferred embodiment, each of said proxy servers has a first processor, a first memory, and a first storage means for storing data; the first memory having at least a logical separation between a first main cache and a first prefetch cache; the content server has a second processor, a second memory, and a second storage means for storing data; and at least one of the clients has a third processor, a third memory, and a third storage means for storing data, the third memory having at least a logical separation between a second main cache and a second prefetch cache, the prefetch handling means storing child objects in the first prefetch cache or the second prefetch cache.

In a preferred embodiment, the prefetching means includes computing means for computing a prefetch value, and evaluating means for evaluating the prefetch value computed by the computing means; said computing means computing a prefetch value for each child object identifier on the prefetch hint information list and prioritizing the caching of child objects with the highest prefetch values on the prefetch hint information list, the prefetch handling means managing available space in the prefetch cache.

Preferably the system further includes: prefetch updating means for updating the prefetch hint information list; the prefetch updating means comparing the first counts to first local threshold counts and second local threshold counts, the first local threshold counts based on statistics from one server, the second local threshold counts based on statistics from another server, said prefetch updating means invoking the

generating means for each of the child object identifiers for which the first counts exceeds both the first local threshold counts and the second local threshold counts; the prefetch updating means keeping the original prefetch hint information list if the first counts do not exceed the first local threshold counts.

It is preferred that such a system further includes:

statistics updating means for updating the statistics table;

a proxy object handler being stored in one of said proxy servers, the proxy object handler invoking a prefetch hint information list update routine, the proxy object handler returning a requested object to a lower level proxy server if the lower level proxy server requested the requested object, the proxy object hander returning the requested object and invoking the statistics updating means if the requested object is not a prefetch object;

a prefetch cache manager for managing a prefetch cache associated with one of said proxy servers, the proxy object handler invoking the prefetch cache manager if the lower level server did not request the requested object;

a proxy object request handler being stored one of said proxy servers, the proxy object request handler returning objects to a requesting one of said clients and marking the object as deletable from the prefetch cache if objects are in the prefetch cache and if the object was requested pursuant to a prefetch request; the proxy object request handler sending an object request to another server if the requested object is not in a main cache or the prefetch cache, the proxy object request handler communicating with the statistics updating means if the requested object is not associated with the prefetch request and if the requested object is located in the prefetch cache or the main cache.

It is also preferred that the system further includes:

statistics updating means for updating the statistics table;

and a prefetch referenced message handler being stored in one of said proxy servers, the prefetch referenced message handler invoking the statistics updating means, the prefetch referenced message handler forwards a prefetch referenced message after invoking a cache manager routine if a requested object is located in the prefetch cache.

The invention further provides a processing system for enhancing a flow of data from servers to clients along a network, the processing system comprising:

proxy servers associated with the network;

a content server associated with the network;

clients configured for communicating with one or more of said proxy servers;

a statistics table being stored in the content server and at least one of said proxy servers, the statistics table storing parent object identifiers and associated child object identifiers, the statistics table containing first counts of parent objects, the first counts representing the frequency in which parent objects are referenced by the clients, the statistics table containing second counts of child objects, the second counts representing the frequency in which the corresponding child objects are proximately, temporally referenced after select ones of said parent objects are referenced;

generating means accepting inputs of first counts, second counts, child object identifiers, and parent object identifiers from the statistics table, the generating means including evaluating means for evaluating the access relationship between parent objects and their associated child objects, the evaluating means placing child object identifiers on a prefetch hint information list if the child objects are referenced with adequate frequencies proximately and subsequently prior to references by the clients to corresponding parent objects; and

prefetching means for prefetching child objects based on the prefetch hint information and parent objects actually requested by the clients.

In a preferred embodiment, the prefetching means further comprises assigning means for assigning storage priorities to child objects identified on the prefetch hint information list and selective storage means for storing select ones of the child objects in accordance with the storage priorities. The assigning means preferably assigns storage priorities based on an estimated access probability that a child object on the prefetch hint information list will be accessed by particular ones of the clients and an access cost of the child object based on requisite storage size for storing select ones of the child objects.

Thus information can be prefetched in a proxy hierarchy to reduce object access time through the network (i.e. Internet) for clients. The select information is prefetched from a content server to make the select information readily accessible to clients associated with a content server in a network. The clients are preferably associated with proxy servers in the network. The clients request information from at least one of the proxy servers. The content server generates prefetch hint information for the clients based on past data access patterns of the requested information by the clients. The content server annotates the requested information with prefetch hint information. One or more proxy servers may

dynamically update the prefetch hint information based on dynamic usage statistics. The prefetch hint information is updated as the object passes through the proxy hierarchy to reflect prefetch operations performed, caching status at higher levels of the proxy hierarchy, and other local considerations, such as local reference patterns. Select information may be prefetched from the content server or any of the proxy servers based upon the prefetch hint information and the requested information. The prefetch hint information may be assigned a prefetch value based upon reference access probability and storage considerations. The select information with the highest prefetch values in the prefetch hint information list are cached first and potentially until the prefetch cache is filled.

A preferred embodiment of the invention will now be described in detail by way of example only with reference to the following drawings:

FIG. 1 is a block diagram showing content servers (i.e. Web sites), a network (i.e. Internet), a proxy server hierarchy, and clients;

FIG. 2 is a further block diagram of a content server, a client, and a proxy server in greater detail than FIG. 1.

FIG. 3A is a flow chart diagram illustrating a method for dynamically prefetching data from a content server via a hierarchy of proxy servers;

FIG. 3B is an illustrative example of a statistics table;

FIG. 4 is a flow chart diagram illustrating operation of the content server;

FIG. 5 is a flow chart diagram illustrating statistics updating;

FIG. 6 is a flow chart diagram illustrating generating of prefetch hint information;

FIG. 7 is a flow chart diagram illustrating client operation;

FIG. 8 is a flow chart diagram illustrating prefetching;

FIG. 9 is a flow chart diagram illustrating operation of a client object request handler;

FIG. 10 is a flow chart diagram illustrating operation of the proxy server;

FIG. 11 is a flow chart diagram illustrating operation of a proxy object request handler;

FIG. 12 is a flow chart diagram illustrating operation of a prefetch referenced message handler;

FIG. 13 is a flow chart diagram illustrating operation of a proxy object handler; and

FIG. 14 is a flow chart diagram illustrating operation of a prefetch hint information update routine.

FIG. 1 shows a data processing system that includes clients 22, a hierarchy of proxy servers 24, a network 25, and content servers 20.

The clients may request information in the form of parent objects, child objects, requested objects, select objects, pages, requested information and select information. Parent objects are related to child objects such that parent objects may have historically proceeded subsequent requests for one or more associated child objects. Requested objects are analogous to parent objects, while select objects are analogous to child objects. Requested objects are objects which are actually requested by one or more clients. Meanwhile, select objects are objects which are grouped with or related to the requested objects such that the select objects may be proximately, temporally requested after the requested objects. Requested information includes requested objects, parent objects, and other forms of data requested by the clients. Select information includes child objects, select objects, and other forms of data that may be prefetched.

The clients 22 may request information from the content servers 20 or proxy servers 24. However, the proxy servers 24 usually act as intermediaries for the clients 22 with respect to the network 25 and content servers 20. Advantageously, the proxy servers 24 may fulfill the clients information requests without contacting the content servers 20, if the proxy server hierarchy contains the requested information or has the ability to complete a requested task. The data processing system of FIG. 1 may be applied to the Internet. For example, the content server 20 may comprise a Web site, while the network 25 may comprise the Internet.

FIG. 1 illustrates logical connections 26, as opposed to physical connections, between elements of the data processing system. The logical connections 26 represent the information flow of objects and requests within the data processing system. The logical connections 26 do not typically represent physical connections between the proxy servers 24. The logical connections 26 can change due to workload and/or physical events such as a node or a communications link failure. Different types of objects may follow different logical paths of transmission to reach the clients 22.

While virtually any number of clients 22 and proxy servers 24 may be used in practice, the illustrative example of FIG. 1 features four clients 22 and four hierarchical levels of proxy servers 24. In practice, any number of levels in the proxy server hierarchy are permitted and the clients 22 can be logically connected to any level of the proxy server

hierarchy. For example, the hierarchy of proxy servers 24 could be a single proxy server. The present invention may be practiced with merely a single client 22 and a single proxy server 24.

As shown, the highest level proxy server is designated as a level-zero proxy server 30. "Level-zero" is an arbitrary designation which merely defines the status of the one proxy server with respect to another proxy server in the illustrative example of FIG. 1. The highest level proxy server may be logically connected to the network 25. The higher level proxy servers 21 are more closely, logically associated with the network 25 than the lower level server proxies 23. (It will be appreciated therefore that in this terminology, which regards the level-zero proxy server as being the top of a proxy hierarchy, the level-zero is "higher" than level-one, which in turn is "higher" than level-two, and so on).

A first level-one proxy server 35 and a second level-one proxy server 37 are logically coupled to the level-zero proxy server 30. The first level-one proxy server 35 is connected to a first level-two proxy server 40 and a second level-two proxy server 43. The first level two proxy server 40 is logically coupled to a first level-three proxy server 50 and a second level-three proxy server 55.

The clients 22 include a first client 600, a second client 603, a third client 700, and a fourth client 703. The first client 600 and the second client 603 are logically coupled to a first level-three proxy server 50. The third client 700 and the fourth client 703 are logically coupled to the second level-three proxy server 55.

The data processing system is now described from the viewpoint of the second client 603. The second client 603 is coupled to its immediately higher level proxy. As illustrated, the immediately higher level proxy is the first level-three proxy 50. The second client 603 is then coupled to the first level-two proxy server 40, the first level-one proxy server 35, and the level-zero proxy server 30, respectively. The second client 603 can access various content servers 20 through the network 25. Although a typical communication path for the second client 603 is via the immediately higher level proxy (i.e. the first level-three proxy server 50), the second client 603 or a lower level proxy may communicate directly to other higher level proxy servers 24 or the network 25. The second client 603 may even have its own client proxy to manage caching.

From the view point of second client 603, certain proxy servers are not part of its proxy hierarchy. For example, the second level-three proxy

server 55 and the second level-two proxy server 43, and the second level-one proxy server 37 are not part of the second client's proxy server hierarchy.

Clients 22 may include, but are not limited to, personal computers, workstations, set top boxes, or the like. The network 25 may include, but does not need to include, the Internet, the World Wide Web, an Intranet, a local area networks (LANs), or the like.

The proxy servers 24 and the content server 20 may comprise various commercially available computers. For example, proxy server 24 may be embodied as the Internet Connection Server (ICS) sold by IBM Corporation. Similarly, the content server 20 may be embodied as a Lotus Go Web server, a Lotus Domino server, or the like (also available from IBM Corporation). The proxy server 24 or content server 20 can run on any computing node, which includes, but is not limited to, products such as S/390 SYSPLEX, SP2, or RS6000 workstations (also available from IBM Corporation). In sum, the proxy server 24 or the content server 20 may comprise any general purpose computer and associated software which can provide World-Wide-Web page accesses, remote file transfers, electronic mail, and transaction support to clients 22.

The data processing system of FIG. 1 may use the attributes of a conventional proxy hierarchy to maintain data transfer efficiency. For instance, if a requested object is not available locally on a lower-level proxy server, a request for the non-available object is made to the next higher-level proxy. If the higher-level proxy has previously cached the object in its cache memory, the higher-level proxy will pass down the requested object to the client via the lower-level proxy server. Otherwise, the higher-level proxy server will try to obtain the object from its next higher level proxy. Once the requested object is obtained, it is passed down to the lower-level proxy or the client requesting the object.

FIG. 2 depicts a more detailed example of the data processing system depicted by FIG. 1. The data processing system includes the proxy server 24, the content server 20, the network 25, and the client 22. The data processing system preferably includes prefetching unit 250, statistics tables 261, statistics updating unit 252, main caches 297, and prefetch caches 298.

The prefetching unit 250 preferably includes a first prefetch handler 248 in the proxy server 24 and a second prefetch handler 241 in

the client 22. Statistics tables 261 include a first statistics table 293 associated with the proxy server 24 and a second statistics table 267 associated with the content server 20. The statistics updating unit 252 includes a first update statistics unit 264 for updating the first statistics table 293 and a second update statistics unit 269 for updating the second statistics table 267. The main caches 297 include a first main cache 294 in each proxy server 24 and a second main cache 246 in each client 22. The prefetch caches 298 include a first prefetch cache 296 in the proxy server 24 and a second prefetch cache 249 in the client 22.

A proxy server 24 is a computing node that can serve requests through the network 25. The proxy server 24 includes a first processor 280, a first memory 290, and a first storage unit 281. The first memory 290 may comprise, for example, random access memory (RAM). The first storage unit 281 may comprise, for example, a direct access storage device (DASD). The first memory 290 stores a proxy server logic 295, which is embodied as a computer executable code. The proxy server logic 295 is loaded from the first storage unit 281 into the first memory 290 for execution by the first processor 280. Details of the operation of the proxy server logic 295 are shown in FIG. 10.

The proxy server logic 295 includes a proxy object request handler 283, a proxy object handler 284, the first prefetch handler 248, a prefetch referenced message handler 285, a prefetch updating means 291, the first update statistics routine 264, a first statistics table 293, a first main cache 294, and a first prefetch cache 296.

Details of the operation of the proxy object request handler 283 are depicted in FIG. 11. Details of the operation of the proxy object handler 284 are shown in FIG. 13. The operation of the prefetch referenced message handler 285 is shown in FIG. 12. Details on the operation of the statistics updating unit 252 (ie the first update statistics routine 264) are shown in FIG. 5.

The operation of the prefetching unit 250 (ie the first prefetch handler 248) is shown in detail in FIG. 8. The operation of the prefetch updating unit 291 or prefetch hint information update routine is shown in FIG. 14. The proxy server logic 295 maintains the first main cache 294 and the first prefetch cache 296.

The content server 20 is a computing node that can serve requests through the network 25. The content server 20 includes a second processor 260, second memory 263, and second storage unit 265. The second memory 263

may comprise, for example, random access memory (RAM). The second storage unit 265 may comprise, for example, a direct access storage device (DASD). The second memory 263 stores the content server logic 268 or content server software.

The operation of the content server logic 268 is shown subsequently in greater detail with reference to FIG. 4. The content server logic 268 is embodied as computer executable code which is loaded from the second storage unit 265 into the second memory 263 for execution by the second processor 260. The content server logic 268 includes a second statistics table 267, a second update statistics unit 269 for updating the second statistics table 267, and generating unit 266 for generating prefetch hint information. The second statistics table 267 stores data on information requested by the clients and potential prefetch information associated with the requested information. The statistics tables 261 contribute to the generation of prefetch hint information which is based on the actual reference behavior of clients. Details on the statistics tables 261 are illustrated in FIG. 3B.

Operation of the second update statistics unit 269 is shown in greater detail with reference to FIG. 5. The generating unit 266 generates prefetch hint information from the information in the second statistics table 267 and the requested information. The generating unit 266 may be embodied as prefetch hint information (PHI) generation routine as depicted in FIG. 6.

The client 22 preferably includes a third processor 240, a third memory 245 and a third storage unit 242. The third memory 245 may comprise, for example, random access memory. The third storage unit 242 may comprise, for example, a direct access storage device (DASD). The third memory 245 stores the client logic 244 or client software. The operation of client logic 244 is subsequently shown in greater detail with reference to FIG. 7. The client logic 244 is embodied as computer executable code which is loaded from the third storage means 242 into the third memory 245 for execution by the third processor 240.

The client logic 244 includes a client object request handler 247, the second prefetch handler 241, a second main cache 246, and a second prefetch cache 249. The client object request handler 247 obtains requested objects for the client from the second main cache 246, the second prefetch cache 249, a proxy server 24, or a content server 20. Details of the operation of client object request handler 247 are shown in FIG. 9.

The client logic 244 maintains the second main cache 246 and the second prefetch cache 249. The second main cache 246 is used for storing frequently referenced objects, while the second prefetch cache 249 is used for storing objects before they actually get referenced. Prefetch objects are objects that have not been proximately or previously referenced yet in conjunction with referenced objects. In the preferred embodiment, the main caches 297 are managed by the conventional LRU (least recently used) algorithm. Those skilled in the art will also appreciate that any other replacement strategy can be applied to maintain the main caches 297.

The main caches 297 and prefetch caches 298 may be logically separated. The logical separation of the prefetch objects from the actually referenced objects in the main cache 297 is to prevent the prefetch objects from taking too much storage space. Various alternatives are available to achieve this objective of a logically separated prefetch cache and main cache. For example, a single cache may have an upper limit imposed on the number of prefetch object and/or an upper limit imposed on the space occupied by the prefetch objects. The main cache 297 and the prefetch cache 298 are meant to support different goals. The main cache 297 is to cache popular objects which have the potential to be repeatedly referenced. The prefetch cache 298 is only used to support a particular browsing session by prefetching objects that are most likely to get referenced next in that session. Under the preferred embodiment, if a lower level node in the hierarchy issues a prefetch request on an object in the prefetch cache of a higher level node, the object will be passed down to the lower level node and be marked as deletable from the prefetch cache 298 of the higher level node since the objective of the prefetch has been accomplished. When a prefetch object actually gets referenced, it may be moved to the main cache 297 depending upon the management policy of the main cache 297. Under the preferred embodiment using an LRU replacement policy, a prefetch object once referenced is moved to the main cache.

An alternative criterion can be devised for admitting an object from the prefetch cache 298 into the main cache 297. In Abrams et al., "Caching Proxies: Limitations and Potentials", Proc. 4th International World Wide Web Conference, 1996, some cache replacement policies have been discussed which do take object size into account in the decision making process. Other cache replacement strategies are known to one of ordinary skill in the art.

FIG. 3A is a flow chart diagram illustrating a method for dynamically prefetching data for clients from a content server 20 or a

proxy server hierarchy. First, in block 100 a client requests information from a proxy server or a content server 20. Second, in block 102 historic data access or data reference patterns at the content server 20 and proxy servers are identified. In practice, the order of completing blocks 100 and block 102 may be reversed.

Third, in block 104 the content server 20 generates prefetch hint information (PHI) on related accesses to the requested information based on the data reference patterns identified in block 102. Fourth, in block 105 the prefetch hint information may annotate the requested information. For example, the prefetch hint information may be conveyed as meta data piggy-backed with the requested information (i.e. object). With respect to the step of establishing or generating prefetch hint information, content servers 20 track usage/reference patterns from a large number of concurrent users or clients. In practice, the content servers 20 determine which group of objects are related to the requested information or requested object. The content servers 20 generate and provide a prefetch hint information (PHI) with the requested object. In the prefetching step or immediately prior thereto, proxy servers may interpret the PHI in light of local considerations to decide which select objects to prefetch.

Fifth, in block 106 the prefetch hint information gets updated as the requested information passes through the proxy server hierarchy. The updating in block 106 reflects prefetch operations performed and caching status at the higher levels hierarchy, and other local considerations such as local reference patterns. The proxy servers may modify the PHI to reflect its prefetch decision and various local considerations before passing the object to the next (lower) level proxy or the client station. As an object passes through the proxy hierarchy, the sending node sets an initial PHI, while the receiving node may derive a final PHI from the initial PHI. The final PHI defines which select objects to prefetch based on the requested objects. The prefetch operation and decision is receiver-initiated, not sender-initiated.

Finally, in block 108 select information in prefetched based on prefetch hint information and the requested information. The proxy server or client decides whether to prefetch a subset of objects specified in the PHI based on the PHI information in the received object, the locally maintained reference information (if available) and other local consideration such as available storage size, and access time.

Generating prefetch hint information in block 104 preferably uses the PICS protocol or a PICS compatible format (see below). A PHI generated in block 104 is stored with each cached object as part of its meta data so that when a cached object is requested, the PHI information of the object can be examined to initiate the appropriate prefetch operations. The PHI information can be provided through the PICS protocol. The PICS protocol provides information on the criticality of each PHI object and its characteristics such as size, and caching status on the higher hierarchy.

Updating the prefetch hint information may be accomplished through the proxy server hierarchy. The proxy servers may collect local information and distribute or decentralize the prefetch decision down the proxy hierarchy. Furthermore, the information of which intermediate proxy has prefetched an object could have implication on the prefetch decision of the lower level proxies and the client station in order to optimize the storage and bandwidth usage.

The proxy servers in the proxy server hierarchy cooperate with each other to provide a receiver-oriented, prefetch decision. The prefetch decision may be made collaboratively by each proxy and client. Prefetch hint information in the format of meta information annotates (i.e. piggy-backs) requested objects. The prefetch hint information provides information on potential prefetch candidates. Intermediate proxy servers may modify the meta information or prefetch hint information based on local conditions, and store the meta information with an object when it is cached, so prefetch can be triggered upon later reference to the object.

FIG. 3B illustrates a data format of the statistics table 261. The statistics table 261 shall refer generally to either the first statistics table 293 or the second statistics table 267. One or more proxy servers 24 maintain the first statistics table 293. The content server 20 may maintain the second statistics table 267. The formats for the first statistics table 293 and the second statistics table 267 are preferably substantially similar or identical to promote updating of the statistics table 261 at different proxy servers in the proxy server hierarchy. The statistics table 261 helps provide prefetch hint information (PHI) based on actual viewer reference behavior from the view point of the proxy server or the content server 20 with which the statistics table 261 is associated. The statistics table 261 is capable of observing the references of all of the communicative clients beneath it in the proxy server hierarchy.

For example, in FIG. 1, the first statistics table 293 in the first level-three proxy server 50 tracks the aggregate data access behavior of the first client 600 and the second client 603. Meanwhile, the first statistics table 293 in the second level-three proxy server 55 tracks the aggregate data access behavior of the third client 700 and the fourth client 703. The first statistics table in the first level-two proxy server 40 tracks the aggregate behavior of the first client 600, the second client 603, the third client 700, and the fourth client 703. Although the lower level proxy, such as the first level-three proxy server 50, may be better able to track the interests of the client 600, the lower level proxy may not be able to get enough reference statistics from a sufficient sample size of clients to provide meaningful prefetch hint information (PHI).

In the preferred embodiment, when an object O is requested from the content server 20, the server 20 provides the initial prefetch hint information PHI based on the information in its second statistics table 267. As the object O passes down the proxy hierarchy, any proxy server 24 along the path to the client 22 can modify the prefetch hint information (PHI) in the object header to tailor the PHI to the local situation in the first statistics table 293, if there are enough local references to the object O. The first statistics table 293 may supersede the second statistics table 267 based upon factors such as the well-indexed interests or preferences of a particular client by a lower level proxy server.

In the preferred embodiment, the content server 20 maintains a second statistics table 267, which is a database of data access frequencies of the clients 22 with respect to requested information. The database could be expanded to include data access frequencies of requesting proxy servers 24. Similarly, one or more proxy servers 24 maintain a first statistics table 293. The statistics table 261, for example, has a first array 383 with a first data structure 304 and a second data structure 306. As shown, the first data structure 304 and the second data structure 306 are records. Each record in the statistics table 261 consists of several fields, including parent object identifiers 301 (i.e. ObjectId), first counts 302 (i.e. TCount), and access list data pointers 303 (i.e. Alist).

The parent object identifier field 301 contains the object identifier. For example the parent object identifier 301 contains the URL of the object for Internet or related applications. The first count field 302 contains the number of times the parent object, associated with the parent object identifier, is referenced as observed by the server. That

is, the first count field 302 contains the frequencies of actual requests for requested objects. The first count 302 represents a parent-object access frequency. The access list data pointer 303 contains a data pointer to an access list 370.

The statistics table 261 has an access list 370, which, for example, contains a first access record 340, a second access record 350, a third access record 360, and a fourth access record 380. As illustrated, the parent object identifier associated with the first data structure 304 has an access list 370 which consists of a linked list of the first access record 340, the second access record 350, and the third access record 360. The next record data pointer field 313 contains a pointer to the next access record on the access list 370 associated with a particular parent object identifier 301. The access list for parent object identifier 301 associated with the second data structure 306 only contains the fourth access record 380.

Each record or data structure on the access list consists of several fields, including a child object identifier 310 (i.e. AObjectId), a second count 312 (i.e. RCount), and a next record data pointer 313 (i.e. Nxt). The data structures may be in the format of general data structure, record, or arrays. As illustrated in FIG. 4, the data structures are records which form a linked list within the access list 370.

In an alternative embodiment, the data format of the statistics table may include a primary array related to a plurality of alternate secondary arrays in a tree structure. The primary array would contain information related to the parent objects and the secondary array would contain information on the child objects of corresponding parent objects.

The child object identifier field 310 contains the object identifier (URL) of an object. The parent object identifier 301 is related to one or more child object identifiers 310 on the access list. The child object identifiers 310 identify the potential prefetch candidates if the child's parent object is actually referenced. The second count 312 contains the number of times the child object is referenced through its parent object. In other words, the second count is the frequency in which the child object is referenced subsequently, temporally and proximately to the prior reference of an associated parent object. The second count 312 represents a child object-access frequency.

The child object identifiers 310 on the access list 370 can be obtained dynamically based on observation of actual reference behavior of the clients. For example, in HTML, a Web page can have HTTP links to

other Web pages. Those Web pages indicated by the HTTP links are potential child objects to the (parent) Web page containing the HTTP links. When the parent object is being viewed, if a reference (HTTP) link contained in the parent object is clicked, the corresponding child object gets referenced. A child object can have more than one parent as a reference link to the child object can be included in multiple other parent objects. Under the HTTP protocol, the request for the child object contains the parent URL as referrer information in its header. So the server can dynamically relate a subsequent child reference to its parent (through the referrer information in the child's header). If a client frequently clicks or selects a child object, the child object may become a prefetch candidate or select information.

In addition to dynamically tracking the viewer reference behavior, static analysis or mining on Web logs can be performed periodically to derive information on groups of objects for the statistics table 261. The mining algorithm identifies groups of objects that are referenced together to obtain potential prefetch candidates for PHI. One method for mining Web logs to derive reference pattern is described in: "SpeedTracer: A Web Usage Mining and Analysis Tool", by Kun-lung Wu, et al., IBM Research Report 20895, May 1997 (see IBM Systems Journal v37, n1, p89-105, 1998). The SpeedTracer algorithm provides information on frequent groups of pages referenced together in a user session and the frequent traversal path. The SpeedTracer algorithm is an in-depth type of mining that can uncover related references that are not immediately referenced together. It can also make better predictions on the next set of referenced objects based on the traversal path to a given object.

Other types of data mining algorithms can be used to identify data access patterns for the statistics table 261. For example, other data mining algorithms include classification and clustering, which can also be applied to the Web log analysis to obtain reference patterns. Furthermore, the PHI can also be supplied by the content provider based on its knowledge of the object relationships. For example, PHI can include, but is not limited to, patches and upgrades for requested software download, Java classes for the requested JAVA applet, and objects (such as GIFs) contained within the requested objects.

In the preferred embodiment, when collecting statistics, actual referencing of an object by a client station is distinguished from a prefetch request to an object. The first count 302 (i.e. TCount) and the second count 310 (i.e. RCount) in the statistics table 261 reflect actual references, not prefetching. That is to say a prefetch operation will not

result in an update to the first statistics table 293 or the second statistics table 267, only actual object references will.

The prefetch hint information or the prefetch hint information list may use various data structures for annotating requested information or requested objects. For example, prefetch hint information may be communicated between requester and server nodes using piggybacked meta data. In a HTTP implementation, the information exchange can be included as meta data in an object header using existing web protocols. PICS ("Platform for Internet Content Selection") specifies a method of sending meta-information concerning electronic content. PICS is applicable to the (a) annotation of the requested information, (b) prefetching select information, prefetch candidates, or child objects, and (c) acknowledging prefetching operations after information is prefetched.

PICS is known to one of ordinary skill in the art as a Web Consortium Protocol Recommendation. PICS was first used for sending values-based rating labels, such as "How much nudity is associated with this content", but the format and meaning of the meta-information is fully general. In PICS, meta-information about electronic content is grouped according to the "rating service" or producer-and-intended-usage of the information, and within one such group, any number of categories or dimensions of information may be transmitted. Each category has a range of permitted values, and for a specific piece of content, a particular category may have a single value or multiple values. In addition, the meta-information group (known as "PICS label") may contain expiration information. There are also facilities for permitting a PICS label to apply to more than one piece of electronic content. Each PICS label for a specific piece of electronic content may be added or removed from the content independently.

For example, an image file may be sent from a server with a PICS label whose "rating service" field indicates it contains values-based rating labels according to the "SafeSurf" rating system. According to the preferred embodiment of the present invention, as it passes through an enterprise proxy, the image file may get a new updated category value for the PICS label to reflect the local consideration of the "rating service". Thus, the client computer will only see the new category value of the PICS label. The HTTP protocol has been augmented so that its request headers and response headers support PICS. The technical bodies which define other common application protocols, such as NNTP, are now also considering adding PICS support. As part of these protocols, a list of the types of PICS labels desired may be included with a request. PICS also specifies a

query format for requesting PICS information from a central label bureau server.

A sample PICS label is:

(PICS-1.1 "http://the.rating.service" labels for "http://the.content" exp "1997.07.09T08:15-0500" r (n 4 s 3 v 2))

where the "n", "s", and "v" are transmit names for various meta-information types, and the applicable values for this content are 4 (for n), 3 (for s), and 2 (for v). Only software which recognizes the ID "http://the.rating.service" would know how to interpret these categories and values.

In a preferred embodiment, three different kinds of PICS labels are introduced. The first kind of PICS label, referred to as a prefetch label (i.e. P-label), is used by the server node to provide PHI. The second kind of PICS label, referred to as a request label (i.e. R-label), is used to request the prefetch candidate objects. The third kind of PICS label, referred to as a reference label (i.e. X-label), is used to provide feedback on the effectiveness of the prefetch, based on which prefetch objects actually get referenced.

The prefetch label can include, but is not limited to, any combination of the following categories including an identifier category, a probability category, a size category, and a caching category. The identifier category (i.e. ID category) has a value that conveys the URL of the prefetch candidate. The probability category (i.e. PROB category) has a value that conveys an estimate of the usefulness of the prefetch. In the preferred embodiment, an estimate of the reference probability is used. The reference probability indicates the probability that the prefetch candidate gets referenced. The size category has a value that conveys the size or memory requirements of the prefetch candidate. The caching category has a value that conveys information on any higher level proxy server that has a copy of the prefetch candidate object either in its first main cache 294 or its first prefetch cache 296. In the preferred embodiment, only the closest higher level proxy is identified.

The request label can include, but is not limited to, the following categories, including the identifier category (i.e. ID category). The value of the identifier category conveys the URL of the object to be prefetched. In the preferred embodiment, when a (proxy or client) node wants to prefetch a suggested prefetch candidate object, a HTTP head request is sent back to the content server 20 through the proxy hierarchy using the request label to convey the URL of the object to be prefetched.

When a proxy server 24 receives a prefetch request from a lower hierarchy node, it will serve the request if the object is in its main or prefetch cache. Otherwise, it forwards the prefetch request to the next higher level proxy.

The reference label can include, but is not limited to, the following categories, including the object identifier category (Object-ID category) and the referrer identifier category (Referrer-ID category). The object identifier category has a value that conveys the URL of the prefetched object that gets referenced. The referrer identifier category has a value that conveys the URL of the object that contains in its PHI the object indicated by the object identifier category value.

In the preferred embodiment, when a prefetch object actually gets referenced, a HTTP head request is sent back to the content server 20 through the proxy hierarchy using the reference label to convey the URL of the prefetch object referenced and its referrer object. This type of HTTP head request is referred to hereinafter as the prefetch referenced message. When a proxy server 24 receives a prefetch referenced message from a lower hierarchy node, it updates its first statistic table 293 and forwards the request to the next higher level proxy server.

FIG. 4 depicts an example of server logic 268 which updates the second statistics table 267 in the content server 20. The updating of the second statistics table 267 is triggered by an object request from a client 22. The content server 20 preferably generates prefetch hint information for annotation or insertion into the object header of the object prior to sending the requested object to the client 22.

Starting at reference block 405, the content server 20 waits for input. In decision symbol 410, if the input received is an object request for object O, the statistics updating unit 252 or second update statistic routine 269 is invoked in reference block 415. In reference block 420, generating unit 266 for generating prefetch hint information 266 or a prefetch generation routine is invoked. A detailed example of the generating unit 266 or PHI generation routine is later described with reference to FIG. 6. In reference block 425, the PHI is, for example, inserted into the object header of the requested object. The prefetch hint information may otherwise annotate the requested object or requested information in block 425. In the preferred embodiment, the PICS protocol may be used to annotate the requested object. The PHI is captured in the P-label, where the identifier category value reflects the URL of the prefetch candidate. In block 430, the requested object is returned to the

requesting client or the requesting proxy server. In decision symbol 435, if the input received is a prefetch referenced message (i.e. a HTTP head request with the X-label indicating that a prefetch object O actually gets referenced) the statistics updating means 252 or the second update statistics routine 269 is invoked in step 440. For other types of inputs, which are not the focus of the present invention, such as an FTP (file transfer protocol) request, an appropriate miscellaneous handler in reference block 445 can be invoked.

FIG. 5 shows how the statistics updating unit 252 updates one of said statistics tables 261. The statistics updating unit 252 refers to the first update statistics routine 264 for updating the first statistics table 293 or the second update statistics routine 269 for updating the second statistics table 267, or both the first update statistics routine 264 and the second update statistics routine 269. The statistics table 261 refers to either the first statistics table 293 or the second statistics table 267, or to both the first statistics table 293 and the second statistics table 267. The statistic updating unit 252 first looks for information in a referrer object. The requested object is then placed on an access list or a prefetch list of the referrer object.

Beginning at reference block 505, the statistics table 261 is updated to reflect that object O has been referenced one more time by incrementing the first count. In other words, in block 505 TCount(O) is incremented by one. If object O is not yet in the statistics table 261, a parent object identifier is inserted into the statistics table 261 with TCount(O) initialized to one. The new parent object identifier replaces the least recently referenced entry in the statistics table 261, if no empty slot is available in the statistics table 261. In decision symbol 510, the requested object is checked for referrer information by, for instance, checking the object header of the requested object. For example, in HTTP protocol, referrer object is provided in the header to indicate which object contains the HTTP link to the requested object. If a referrer object (R) is found in the header, in step 520 the statistics table 261 is examined to see whether object O is already on the access list of object R. If so, the second count (i.e. RCount(O)) is incremented by one in block 530 for the requested object. Otherwise, in block 525 the requested object (i.e. Object O) is inserted into the access list of the parent object (i.e. object R) with the second count (i.e. RCount(O)) initialized to one.

FIG. 6 depicts an example of the generating unit 266 for generating prefetch hint information into a prefetch hint information list. The

prefetch hint information list contains child object identifiers of child objects if the ratio of the second count to the first count exceeds a hint threshold. The prefetch hint information list represents a refined or filtered access list. That is, the generating unit 266 accepts inputs from the statistics table 261 and filters the inputs to obtain the prefetch hint information list.

The generating unit 266 for generating prefetch hint information represents a prefetch hint information (PHI) generation routine 266, which begins in reference block 610. In reference block 610, the access list of a parent object (i.e. object O) is checked to see whether or not it is empty (i.e. whether the Alist(O) is null). If the access list is not empty, in reference block 620, let Cj be the next child object (i.e. child object identifier) identified on the access list of parent object (i.e. object O).

In decision symbol 630, the reference probability of the child object Cj is compared with a hint threshold (i.e. TH), which is a pre-specified threshold value (eg 0.75). In the preferred embodiment, the reference probability of Cj is defined to be the ratio of the second count to the first count (i.e. RCount(Cj)/TCount(O)). The ratio is an estimate of the probability that child object (object Cj) gets referenced after the parent object (object O) is referenced. If in decision symbol 630, the reference probability of Cj is larger than the hint threshold (i.e. TH), then the child object identifier (Cj) will be included in the PHI list in reference block 640. In other words, if the ratio exceeds the hint threshold, the child object identifier and associated information is placed on the prefetch hint information list.

The generating unit 266 may require software subroutines or software components to practice the comparison in reference block 630. In particular, the generating unit 266 for generating a prefetch hint information into a prefetch hint information list may include calculating means to calculate the reference probability and comparing means to compare the reference probability to the hint threshold.

In the preferred embodiment, using a PICS protocol, the PHI information is stored in the P-label, where the URL of child object (Cj) is taken as the identifier category (i.e. ID category value), its reference probability and size are taken as the probability category (i.e. PROB) and size category values, respectively. In decision symbol 645, if the access list of object O is not yet fully scanned, reference block 620 will be re-executed.

Various alternative criterion can be used to select the prefetch candidates from the statistics table 261. For example, a moving average of the reference probability over past intervals can be used as the criterion. Furthermore, the selection procedure of the prefetch candidates can be made recursive. If a child object (object Cj) is selected as a prefetch candidate of parent object (object O), any child object (object Cji) on the access list of the child object (object Cj) may be evaluated to determine whether to be included in the prefetch list of the parent object (object O), if not yet included. At decision symbol 630, the comparing unit or evaluating unit may now test whether the product of RCount(Cj)/TCount(O) and RCount(Cji)/TCount(Cj) is larger than the hint threshold (TH).

FIG. 7 depicts an example of the operation of client logic 244. FIG. 7 shows that once an object is requested, the client object request handler 247 is invoked. Otherwise, the prefetch handler 241 may handle objects which are not requested, but need to be stored in the second prefetch cache 249 of the client 22.

Beginning in block 710, the client 22 waits for input. In decision symbol 720, if the input received is an object, the second prefetch handler 241 is invoked in reference block 730. In block 740, the client object handler is invoked to return the object to the requesting application at the client 22. If in decision symbol 750 the input received is an object request, the client object request handler 247 is invoked at reference block 760. For other types of inputs, which are not the focus of the present invention (such as a push request) an appropriate miscellaneous handler at reference block 770 can be invoked.

FIG. 8 depicts an example of the operation of the prefetching unit 250, including the first prefetch handler 248, or the second prefetch handler 241, or both the first prefetch handler 248 and the second prefetch handler 241. To summarize, the prefetching unit 250 uses the prefetch list generated by the generating unit 266 in FIG. 6 as an input to determine if child objects referenced on the prefetch hint information list warrant storing in one of the prefetch caches 298. The prefetching unit 250 calculates a prefetch value for each prefetch object candidate or child object identifier on the prefetch hint information list. The prefetch value considers access probability of the child object and access cost of storing the child object. The child objects with the highest threshold values are cached first until the available space in one of the prefetch caches 298 is filled.

Starting in reference block 805, let L be the PHI list of prefetch candidate objects. The prefetch candidate objects may be expressed in terms of child object identifiers on the prefetch hint information list. Any prefetch candidate appearing in the local cache (i.e. prefetch cache or main cache) is removed from L. In block 810, the prefetch value (PV) of each object in L is calculated. In the preferred embodiment, the PV value is defined to be the reference probability multiplied by the access cost. The access cost is estimated based on the closest site caching the prefetch candidate object or the child object identified on the PHI list. As a practical matter, the child objects preferably are identified on the PHI list by child object identifiers rather than as the entire child objects to conserve storage space. The reference probability optimally comes from the probability category (PROB category) value and the closest caching site comes from the caching category value of the PICS label. In block 820, the prefetch candidate object with the highest PV value is selected. In reference block 830, the amount of prefetch buffer space available to accommodate the prefetch object candidate (i.e. Oj) is determined. This space is either not currently in use or occupied by objects with lower PV values which are only a fraction (e.g. one half) of the PV value of the present prefetch object candidate being evaluated (i.e. Oj). Any storage space occupied by an object marked as deletable in one of the prefetch caches 298 is considered not currently in use. In decision symbol 840, one of the prefetch caches 298 is checked to see whether there is enough space for Oj. If so, in reference block 850, object Oj is prefetched into one of the prefetch caches 298. In block 860, Oj is removed from the PHI list (i.e. L). In reference block 870, if L is not empty, the step associated with reference block 820 is re-executed.

Although in the preferred embodiment each object is prefetched one at a time, multiple objects can be prefetched together in one request. In reference block 850, Oj can be put onto a prefetch list. In block 870, if L is empty, a prefetch request is then issued with the whole prefetch list. Furthermore, even if there is not enough storage space in the prefetch cache, a pending prefetch list can be maintained, so that when storage becomes available, additional prefetch can be initiated.

By using the HTTP protocol, the communication session can remain open during the prefetch to reduce communication overhead. There is no need to reestablish the session for each prefetch.

FIG. 9 depicts an example of the operation of the client object request handler 247. To summarize, the client object request handler 247

first searches the second main cache 246 and secondly searches the second prefetch cache 249. If the object is in neither the second main cache 246 nor the second prefetch cache 249, an object request is sent to the server. If the object is in the second prefetch cache 249 a prefetched referenced message is sent to one of the servers to indicate the prefetch object was actually referenced. Then, the second prefetch handler 241 and the client object handler are used.

Starting at decision symbol 905, if the requested object O is found in the second main cache 246, the second prefetch handler 241 is invoked in reference block 925 to initiate appropriate prefetch operations based on the PHI associated with the object O. In reference block 930, the client object handler is invoked to return the requested object to the requesting application at the client. In decision symbol 910, if the requested object O is found in the second prefetch cache 249, a prefetch referenced message is sent to the content server 20 in block 920 so that its second statistics table 267 can be updated to capture the actual referencing of the prefetch object. In fact, as the prefetch referenced message propagates through the proxy hierarchy to the content server 20, the second statistics table 267 in each intermediate proxy will also get updated. In block 910, if the object is not in the second prefetch cache 249, an object request is send to the content server 20 through the proxy hierarchy in reference block 915.

The prefetch referenced messages can be batched. That is, a prefetch referenced list can be maintained and periodically a composite prefetch referenced message is sent to the higher level server to acknowledge the referencing of all objects on the prefetch referenced list.

FIG. 10 depicts an example of the operation of a proxy server logic 295. In summary, the input is sorted in accordance with whether the input is an object request, an object transmission, or a prefetch referenced message. Three different handlers or software routines are available depending upon the input: a proxy object request handler 283, a proxy object handler 284, and a prefetched reference message handler 285.

Starting at reference block 1005, the proxy server 24 waits for input. If in decision symbol 1010 the input received is an object request for an object O, the proxy server 24 checks whether there is already a pending request on that object O in block 1015. The pending request may be due to a prefetch request on that object O issued by this proxy server 24 or an earlier request to the object O originated from another proxy

server node, which is lower in the proxy server hierarchy. If there is no pending request on the object O in reference block 1015, the proxy object request handler 283 is invoked in block 1020. In decision symbol 1025, if the input received is an object (from the higher level proxy), the proxy object handler 284 is invoked in reference block 1030. Otherwise, in block 1040, if the input received is a prefetch referenced message, the prefetch referenced message handler 285 is invoked in block 1045. For other types of inputs, which are not the focus of the present invention (such as the an FTP request) an appropriate miscellaneous handler in block 1060 can be invoked.

FIG. 11 depicts an example of the operation of the proxy object request handler 283. To summarize, if objects are in the prefetch cache, the object is returned or sent to the requesting client or requesting proxy server. Thereafter, if the requested object was a prefetch request, the requested object is marked as deletable from the first prefetch cache 296 of the proxy server 24. If no prefetch request was provided for the requested object, a prefetch referenced message is sent to the content server 20. However, the prefetched reference message is only sent when the requested object is sent from the first prefetch cache 296. If a requested object is neither in the first prefetch cache 296 nor the first main cache 294, an object request is sent to another proxy server 24 or a content server 20. Requested objects found in the first main cache 294 are returned or sent to the requesting client or proxy server 24 and the first statistics table 293 is updated if the request was not a prefetch request.

Beginning in decision symbol 1105, if the requested object O is found in the prefetch cache, the object O is returned to the requesting node in block 1130. In decision symbol 1135, if the request is a prefetch request, the object O in the first prefetch cache 296 is marked deletable in block 1150. Deleting objects from the first prefetch cache 296 makes room for future prefetch because the prefetch object O has now been prefetched by some nodes in the lower hierarchy of proxy servers 24. Otherwise, in reference block 1140, the cache manager is invoked to move the object O from the first prefetch cache 296 to the first main cache 294 since the object O has actually been referenced. In block 1160, a prefetch referenced message is sent to the content server 20 through the proxy hierarchy server. In block 1170, the update statistics routine is invoked to update the local first statistics table 293 to capture the actual referencing of the prefetch object O. In decision symbol 1110, if the object O is not found in the first main cache 294, an object request is send to the content server 20 through the proxy server hierarchy in

reference block 1125. In decision symbol 1115, if the object request is not a prefetch request, the first update statistics routine 264 is invoked in block 1120 to update the local first statistics table 293 to capture the actual referencing of the object. In block 1180, the object O is returned to the requesting node.

FIG. 12 depicts an example of the operation of the prefetch referenced message handler 285. The prefetch referenced message handler 285 first looks at whether an object is in the first prefetch cache 296 or not. If the object is in the first prefetch cache 296, the cache manager routine is invoked prior to forwarding the prefetch referenced message. If the object is not in the first prefetch cache 296, the prefetch referenced message is forwarded.

Beginning at step 1205, the first update statistics routine 264 is invoked to update the local first statistics table 293 to capture the actual referencing of the prefetch object (O). In decision symbol 1210, if the object O is in the first prefetch cache 296, the cache manager is invoked in reference block 1215 to move the object O from the first prefetch cache 296 to the first main cache 294 since the object has actually been referenced. In block 1220, the prefetch referenced message is forwarded to the next higher level proxy.

FIG. 13 depicts an example of the operation of the proxy object handler 284. The proxy object handler 284 first invokes the prefetch updating unit 291 or a prefetch hint information update routine uniformly without exception. Prefetch object requests are handled differently depending upon whether the prefetch object was requested by a lower level server or not. If the prefetch object was requested by a lower level server, the object is returned to the lower level server; otherwise the prefetch cache manager is invoked. If a prefetch object is not requested, the object is returned to the requesting proxy server 24 or client and the first update statistics routine 264 is invoked.

Beginning at reference block 1305, the PHI update routine or the prefetch updating means 291 is invoked to update the PHI of object O. In reference block 1310, if the object received is not a prefetch object, the first prefetch handler 248 is invoked in step 1315. In reference block 1320, the object is returned to the requesting node (in the lower hierarchy). Under the preferred embodiment using the PICS protocol, the caching category value in the P-label is updated if the corresponding prefetch candidate (indicated by the identifier category value) is prefetched by this proxy server 24. In block 1325, the cache manager is invoked to store the object in the main cache. In reference block 1355,

the update statistics routine is invoked. In block 1330, if the prefetch object received is requested by a lower level node in the hierarchy, the object is returned to the requester in reference block 1350. Otherwise, it is requested by the current node, and the prefetch cache manager is invoked in reference block 1340 to store the object in the prefetch cache by making necessary replacement to make room for the object.

FIG. 14 depicts an example of the prefetch updating means 291 or the PHI update routine. The PHI update routine compares a local count of requested objects to a threshold count. If the requested objects exceed a first minimum threshold, they are subjected to a second minimum threshold test. The thresholds may be set based upon experimental results. If the local count is less than the first minimum threshold, the original prefetch hint information list is kept. If the local count is greater than the first threshold, but not greater than the second threshold, the prefetch hint information list is modified.

Beginning in decision symbol 1405, if the local count (i.e. TCount) of the object (O) in the local first statistics table 293 is not larger than a first minimum threshold (i.e. CTM), the original PHI in the header of the object O is kept. This is simply due to the fact that there is not enough referencing of the object in the lower hierarchy of proxy servers 24 to warrant any updating action. In decision symbol 1415, if the local count (i.e. TCount) of the object (O) in the local statistics table 261 is larger than a second minimum threshold (i.e. TCH), the original PHI in the object header is ignored and the PHI generation routine is invoked in block 1420 to generate the PHI based on the local first statistics table 293. This is because there are sufficient references to the object lower in the hierarchy of proxy servers 24 to generate the PHI based on the local behavior. Otherwise, the PHI modifier is invoked in reference block 1425 to modify the probability category value by taking an average of the local estimate of the reference probability and that in the header (set by the higher level hierarchy) for each object in the PHI list.

The prefetch scheme works even if some of the proxy servers 24 do not maintain the first statistics table 293 and track local reference patterns. The non-tracking proxy servers simply accept the probability (i.e. PROB) category value in the PHI passed down from the higher level proxy server or content server 20.

The prefetch scheme also works in a heterogeneous proxy-server environment, where some of the proxy servers are conventional proxy servers which do not understand the prefetch protocol and do not

participate in the collaboration. These conventional proxy servers will neither interpret the object header to initiate the prefetch operations, nor maintain the first statistics table 293. The conventional proxy servers will only pass the meta data (e.g. the PICS labels in the header) with the object to the next level proxy server or store the meta data with the object if it decides to cache the object locally.

Even without the presence of the proxy hierarchy, the current invention can be applied to any network or client server environment. The object sender node can track reference patterns, derive PHI and include it in the object header. Furthermore, the object receiver node can store the PHI with the object in its cache. Upon referencing of the object, prefetch operations can be initiated based on the stored PHI and local factors such as cache size and access cost as discussed before.

Storing the PHI with the cached object provides additional opportunity for prefetching. Events such as more storage becoming available, additional references occurring or prefetch objects being received, can be used to trigger additional prefetch by examining the PHI of the cached objects. The prefetch operation is not restricted to the moment that an object is requested from the server as in the sender initiated approach.

In the preferred embodiment, a general prefetch policy for web and proxy servers has been described, but those skilled in the art will appreciate, however, that the present invention is applicable to any kind of situation where the objects to be prefetched have similar characteristics, and is not necessarily restricted to an Internet or WWW application.

Further, while the preferred embodiment of the present invention is for collaborative prefetching between parent and child proxies in the hierarchy, it can be easily adapted to include collaboration between sibling proxies. For example if no proxy in the upper hierarchy has cached the requested object or prefetch candidate, a proxy can also interrogate sibling proxies.

CLAIMS

1. A method for prefetching select information from a content server (20) to make the select information readily accessible for one or more clients (22) associated with the content server via a communications network (25) and one or more proxy servers (24); the method comprising the steps of:

requesting (100) information from at least one of the proxy servers;

establishing (102, 104) prefetch hint information for said clients based on past retrievals or accesses of the requested information by any of said clients;

annotating (105) the requested information with said prefetch hint information; and

prefetching (108) select information from any of said servers based upon the prefetch hint information and the requested information.

2. The method of claim 1, wherein the requested information comprises requested objects, and wherein the select information comprises prefetch candidate objects.

3. The method of claim 2, wherein the step of establishing prefetch hint information includes the steps of:

tracking data reference patterns associated with said clients;

organizing the data reference patterns into groups of object identifiers, the groups of object identifiers organized according to relationships between the requested objects and prefetch candidate objects;

determining which of said groups have sufficient relationships between the prefetch candidate objects and the requested objects to warrant prefetching of said prefetch candidate objects as the select information.

4. The method of claim 3, wherein the steps of tracking and organizing the data reference patterns include the steps of:

establishing databases for storing reference data patterns in the proxy servers;

arranging fields within each of said databases, the fields including object identifiers and object counts, the object identifiers distinguishing the requested objects from one another; the object counts indicating the number of times the requested objects were requested as observed by one of said servers; and

associating access lists with said databases, each of said access lists containing one or more prefetch candidate objects associated with each requested object.

5. The method of claim 3 or 4, wherein the steps of tracking and organising the data reference patterns further comprise the steps of:

establishing a database defining a relationship between requested objects and prefetch candidate objects;

tracking access statistics on the requested objects and the prefetch candidate objects; the access statistics including a parent-object access frequency of requested objects and a child-object access frequency of the prefetch candidate objects; the parent-object access frequency representing the number of times in which a particular one of the requested objects is requested by at least one of said clients; the child-object access frequency representing the number of times in which one or more of said prefetch candidate objects are actually requested proximately following the accessing of the particular one.

6. The method of claim 5, further comprising the step of selecting a prefetch list of prefetch candidate objects based upon the ratio of the child-object access frequency to the parent-object access frequency associated with the child-object access frequency, the prefetch list containing only prefetch candidate objects with ratios that exceed a threshold probability.

7. The method of any of claims 2 to 6, wherein the step of prefetching select information comprises the steps of:

prefetching a subset of prefetch candidate objects received at one of said proxy servers or one of said clients;

selecting the subset of the prefetch candidate objects as select information based on available storage size at one of said proxy servers.

8. The method of any of claims 2 to 7, wherein the step of prefetching select information comprises the steps of:

prefetching a subset of prefetch candidate objects received at one of said proxy servers or one of said clients;

selecting the subset of the prefetch candidate objects as select information based on access time at one of said proxy servers.

9. The method of any of claims 2 to 8, wherein the select object and the requested object comprise Internet data and wherein the requested object references the select object through a hypertext transfer protocol (HTTP) link associated with the Internet data, the proxy server

identifying the relationship of the select object to the requested object through the HTTP protocol of the requested object universal resource locator, or URL, stored in an object header of the requested object.

10. The method of any of claims 2 to 9, wherein the step of annotating the requested information comprises the step of placing the prefetch hint information in a prefetch label having a Platform for Internet Content Selection (PICS) format, or a label protocol compatible with the PICS format.

11. The method of claim 10, wherein the Platform for Internet Content Selection (PICS) format includes an object identifier category and a referrer identifier category, and said method further includes the step of returning the referrer identifier category to the content server to identify any prefetch candidate objects which were actually referenced.

12. The method of claim 1 or 2, wherein the step of establishing prefetch hint information includes the steps of:

identifying a data reference pattern of said clients based upon the past retrievals or prior accesses of the requested information, the data reference pattern selected to represent any common trends in past retrievals or accesses by different ones of said clients associated with one of said proxy servers; and

defining prefetch hint information from the data reference pattern.

13. The method of any preceding claim, there is a hierarchy of said proxy servers, the hierarchy containing a sending node and a receiving node, and said method further comprises the steps of:

transmitting prefetch hint information from the sending node to the receiving node; and

modifying the prefetch hint information from an initial state to a final state, the initial state determined by the sending node, the final state determined by the receiving node; the prefetch hint information in said final state being designated as an ultimate prefetch decision.

14. The method of claim 13, wherein the step of prefetching select information is initiated by the receiving node.

15. The method of any of claims 1 to 12, further comprising the steps of:

organizing the proxy servers into a hierarchy of servers, the hierarchy having multiple levels of the proxy servers, the multiple levels

being determined based on the proximity of the each of said proxy servers to the network, with the highest level being closest to the network; and

interpreting and updating the prefetch hint information transmitted through the hierarchy from the proxy servers to the clients.

16. The method of claim 15, wherein a proxy server updates the prefetch hint information if local references to the requested information exist and if the count of the local-level server exceeds a minimum threshold count.

17. The method of claim 15 or 16, wherein the prefetch hint information is updated based on caching status in the proxy hierarchy.

18. A data processing system for prefetching select information from a content server (20) over a network (25), said system including one or more proxy servers (24) associated with the network, and clients for communicating with the content server and one or more of the proxy servers over the network; said system comprising:

means for requesting information from at least one of the proxy servers;

means for establishing prefetch hint information for said clients based on past retrievals or accesses of the requested information by any of said clients;

means for annotating the requested information with said prefetch hint information; and

means for prefetching select information from any of said servers based upon the prefetch hint information and the requested information.

19. The processing system of claim 18, wherein the requested information comprises requested objects, and wherein the select information comprises prefetch candidate objects, and wherein each of said proxy servers and at least one of the clients has a memory having at least a logical separation between a main cache and a prefetch cache, wherein the prefetched select information is stored in the prefetch cache.

20. The processing system of claim 19, wherein the prefetching means prefetches objects and stores the prefetched objects in the prefetch cache based on a priority scheme, the priority scheme storing objects with a highest prefetch value first, with regard to the available memory of the prefetch cache, the prefetch value being based upon an individual or aggregate reference probability to the prefetch objects by one or more clients.

21. The processing system of claim 20, wherein the prefetch value is based on an estimated access probability that a prefetch candidate object on the prefetch hint information list will be accessed by particular ones of the clients and an access cost of the object based on requisite storage size for storing select ones of the prefetch candidate objects.

22. The processing system of claim 20 or 21, wherein the prefetching means comprises a prefetch handler including computing means for computing said prefetch value, and evaluating means for evaluating the prefetch value computed by the computing means; said computing means computing the prefetch value for each prefetch object identifier on the prefetch hint information list; said evaluating means evaluating the prefetch values of multiple candidate prefetch objects and prioritizing the caching of candidate prefetch objects associated with the highest prefetch values on the prefetch hint information list.

23. The processing system of claim 22, wherein the prefetching means comprises a first prefetch handler in at least one of said proxy servers and a second prefetch handler in at least one of said clients.

24. The processing system of any of claims 19 to 23, further including a client object request handler stored in the memory of one of said clients, the client object request handler searching the main cache and prefetch cache for a requested object prior to requesting the requested object from a proxy server or the network, the client object request handler notifying the content server or proxy server if the object is located in the prefetch cache by generating a prefetch referenced message to the server indicative that the prefetch object was actually accessed by the client.

25. The processing system of any of claims 19 to 24, wherein at least one of said proxy servers includes a prefetch referenced message handler for forwarding a prefetch referenced message after invoking a cache manager routine if a requested object is located in the prefetch cache.

26. A content or proxy server for use in the data processing system of any of claims 18 to 25, wherein the requested information comprises requested objects, and wherein the select information comprises prefetch candidate objects, said server including:

a statistics table storing parent object identifiers and associated child object identifiers, the statistics table containing first counts of parent objects, the first counts representing the frequency with which parent objects are referenced by the clients, the statistics table containing second counts of child objects, the second counts representing

the frequency with which the corresponding child objects are referenced after select ones of said parent objects are referenced; and

generating means for generating the prefetch hint information list based on information in the statistics table.

27. The server of claim 26, wherein the generating means accepts inputs of the first counts and the corresponding second counts to derive the prefetch hint information list, said generating means including: calculating means for calculating a ratio of the second count to the first count for each of said parent object identifiers; and comparing means for comparing the ratio to a hint threshold, the comparing means grouping or associating child objects and associated parent objects on the prefetch hint information list if the ratio exceeded the hint threshold.

28. The server of claim 26, wherein the generating means accepts inputs of first counts, second counts, child object identifiers, and parent object identifiers from the statistics table, the generating means including evaluating means for evaluating the access relationship between parent objects and their associated child objects, the evaluating means placing child object identifiers on a prefetch hint information list if the child objects are referenced with adequate frequencies proximately and subsequently prior to references by the clients to corresponding parent objects.

29. The server of claim 28 wherein the generating means further comprises calculating means for calculating the adequate frequencies based on the ratio of child accesses to corresponding parent accesses.

30. The server of any of claims 26 to 29, further comprising:

statistics updating means for updating the statistics table based upon a child object actually requested; the statistics updating means first searching for an identity of a referring one of said parent objects, the referring one, or referrer object, being referenced prior to the child object actually requested, the child object actually requested having its corresponding second count updated in the statistics table.

31. The server of claim 30, wherein the statistics updating means updates the statistics table upon receipt of a prefetch referenced message from a prefetch referenced message handler.

32. The server of any of claims 26 to 31, wherein said server is a proxy server and further comprises:

prefetch updating means for receiving and updating the prefetch hint information list; the prefetch updating means comparing the first counts to first local threshold counts, said prefetch updating means invoking the generating means for each of the child object identifiers for which the first counts exceeds the first local threshold counts, the prefetch updating means keeping the original prefetch hint information list if the first counts do not exceed the first local threshold counts.

33. The server of claim 32, wherein the prefetch updating means compares the first counts to first and second local threshold counts, and (i) keeps the original prefetch hint information list if the first counts do not exceed the first local threshold counts; (ii) modifies the original prefetch hint information list if the first counts exceed the first local threshold counts but not the second local threshold counts; and (iii) replaces the original prefetch hint information list if the first counts exceed both the first and second local threshold counts.

34. The server of claim 32 or 33, further including a proxy object handler for returning a requested object to a lower level proxy server or client that requested the requested object, the proxy object hander returning the requested object and updating the statistics table accordingly if the requested object is not a prefetch object.

35. The server of claim 34, further including:

a prefetch cache manager for managing a prefetch cache;

wherein the proxy object handler returns a requested object from the prefetch cache to a requesting client or proxy server, and marks the object as deletable from the prefetch cache if the object was requested pursuant to a prefetch request; the proxy object handler sending an object request to another server if the requested object is not in a main cache or the prefetch cache, the proxy object handler updating the statistics table accordingly if the requested object is not associated with a prefetch request and if the requested object is located in the prefetch cache or the main cache.

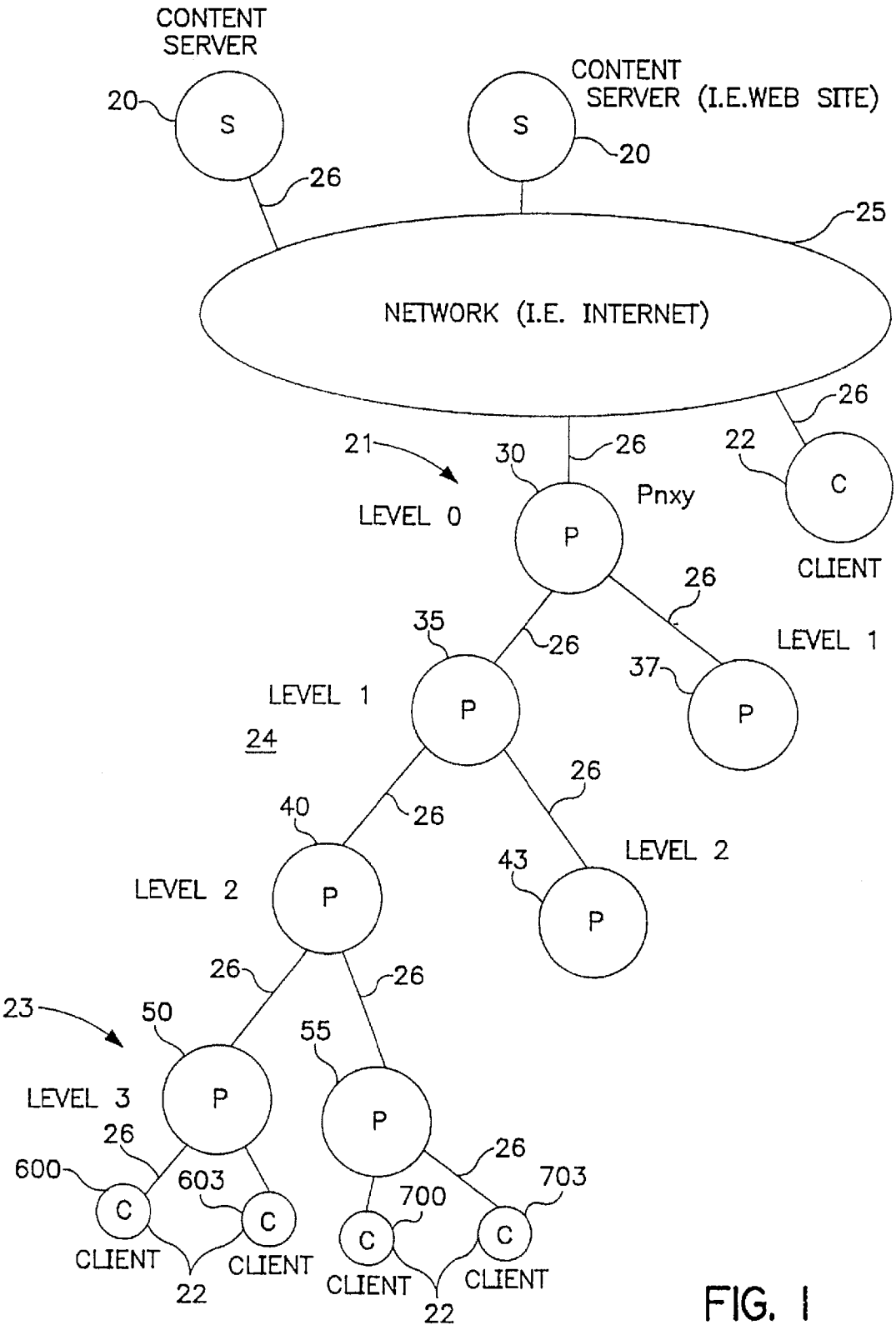

CONTENT SERVER
20
S
26
CONTENT SERVER (I.E.WEB SITE)
S
20
25
NETWORK (I.E. INTERNET)
26
22
C
CLIENT
21
LEVEL 0
30
26
Pnxy
P
26
35
26
LEVEL 1
37
LEVEL 1
P
P
24
26
40
26
LEVEL 2
43
LEVEL 2
P
P
26
26
23
50
55
LEVEL 3
P
P
26
26
600
603
700
703
C
C
C
C
CLIENT
CLIENT
CLIENT
CLIENT
22
22

FIG. 1

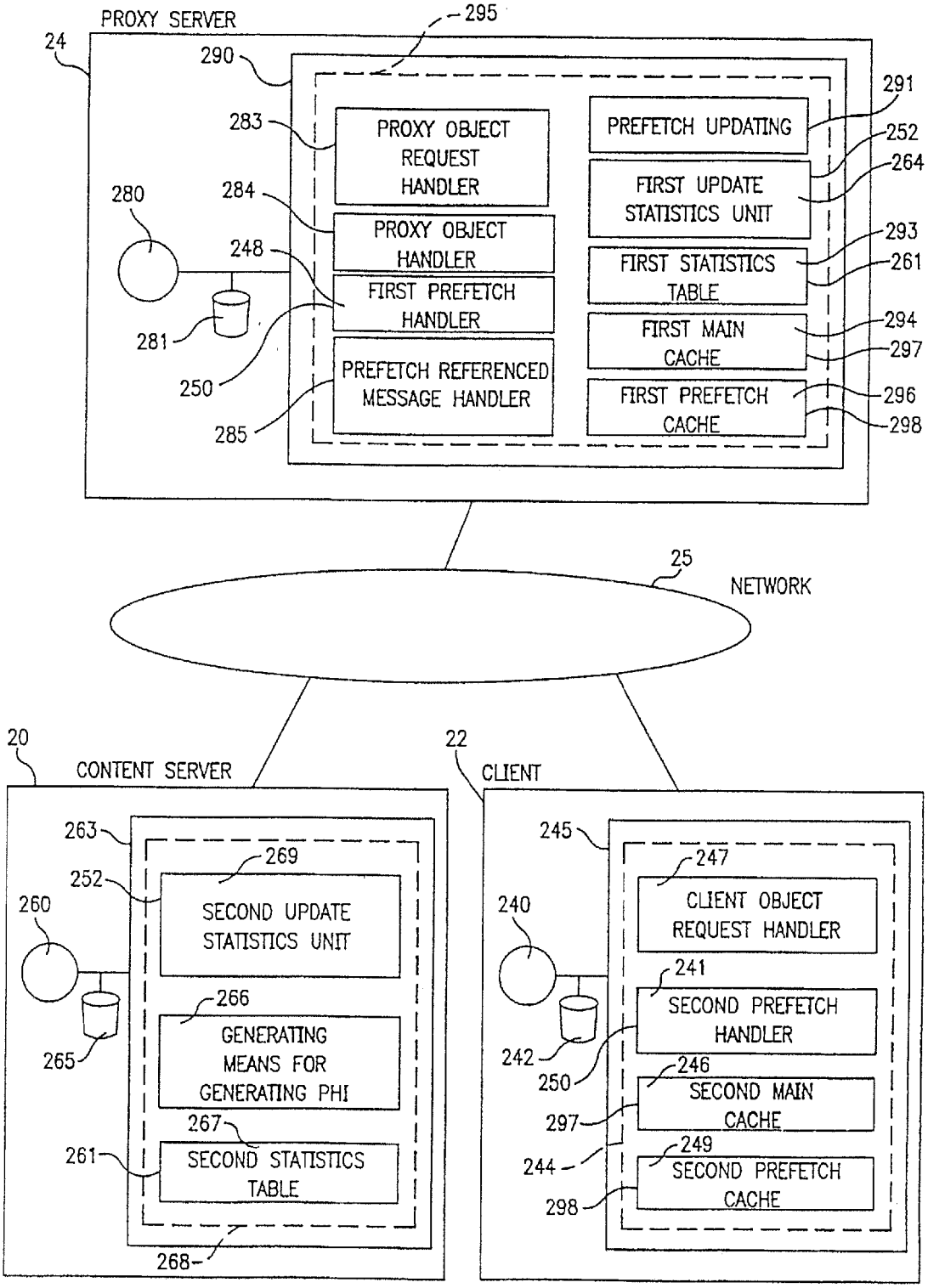

PROXY SERVER
24
295
290
291
283
PROXY OBJECT REQUEST HANDLER
PREFETCH UPDATING
252
264
280
284
PROXY OBJECT HANDLER
FIRST UPDATE STATISTICS UNIT
248
293
261
FIRST PREFETCH HANDLER
FIRST STATISTICS TABLE
281
294
297
FIRST MAIN CACHE
250
PREFETCH REFERENCED MESSAGE HANDLER
296
298
FIRST PREFETCH CACHE
285
25
NETWORK
20
CONTENT SERVER
22
CLIENT
263
269
252
SECOND UPDATE STATISTICS UNIT
260
245
247
CLIENT OBJECT REQUEST HANDLER
240
241
266
GENERATING MEANS FOR GENERATING PHI
SECOND PREFETCH HANDLER
265
242
250
246
SECOND MAIN CACHE
267
297
249
261
SECOND STATISTICS TABLE
244
SECOND PREFETCH CACHE
298
268

FIG. 2

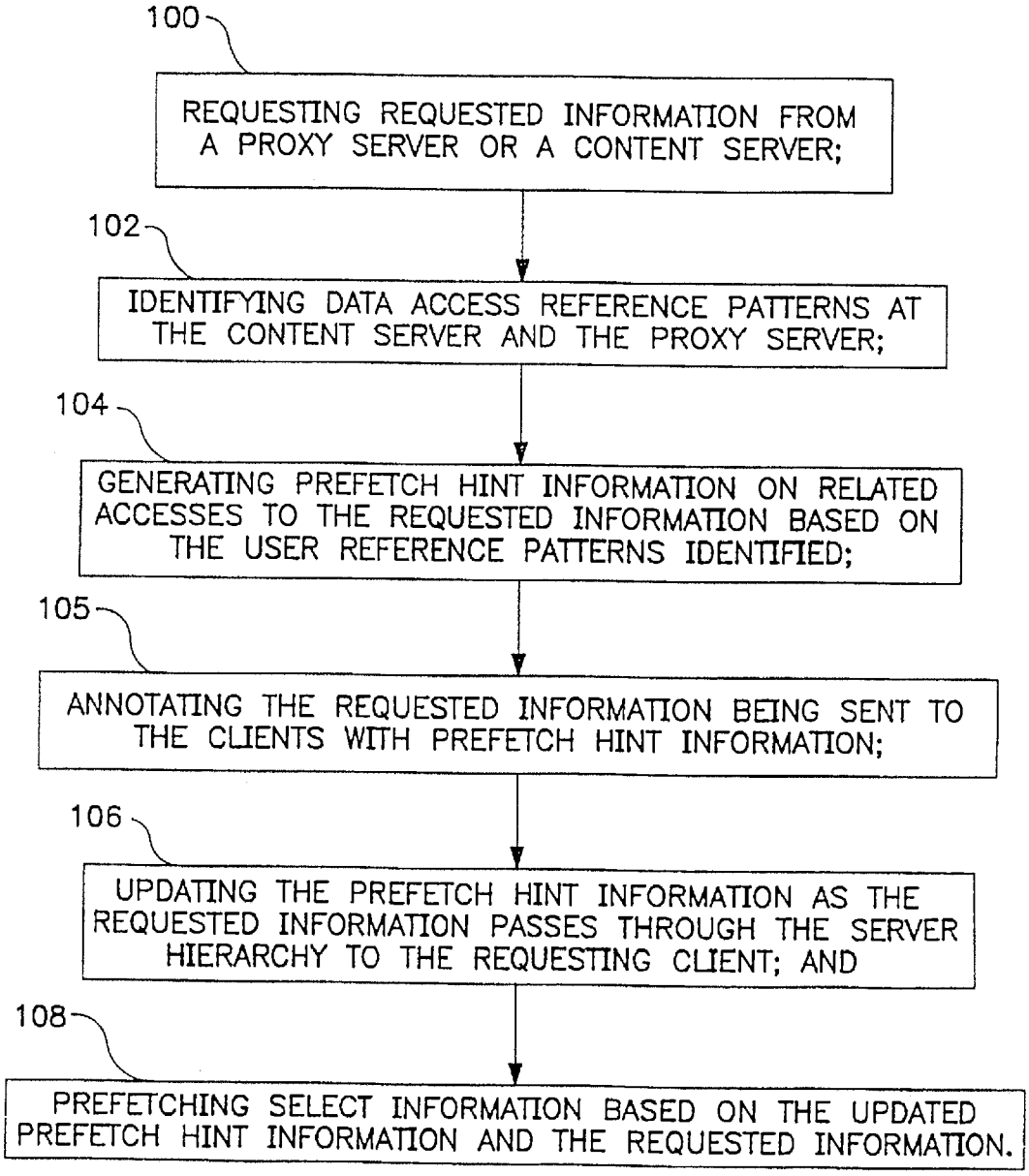

100
REQUESTING REQUESTED INFORMATION FROM A PROXY SERVER OR A CONTENT SERVER;
102
IDENTIFYING DATA ACCESS REFERENCE PATTERNS AT THE CONTENT SERVER AND THE PROXY SERVER;
104
GENERATING PREFETCH HINT INFORMATION ON RELATED ACCESSES TO THE REQUESTED INFORMATION BASED ON THE USER REFERENCE PATTERNS IDENTIFIED;
105
ANNOTATING THE REQUESTED INFORMATION BEING SENT TO THE CLIENTS WITH PREFETCH HINT INFORMATION;
106
UPDATING THE PREFETCH HINT INFORMATION AS THE REQUESTED INFORMATION PASSES THROUGH THE SERVER HIERARCHY TO THE REQUESTING CLIENT; AND
108
PREFETCHING SELECT INFORMATION BASED ON THE UPDATED PREFETCH HINT INFORMATION AND THE REQUESTED INFORMATION.

FIG. 3A

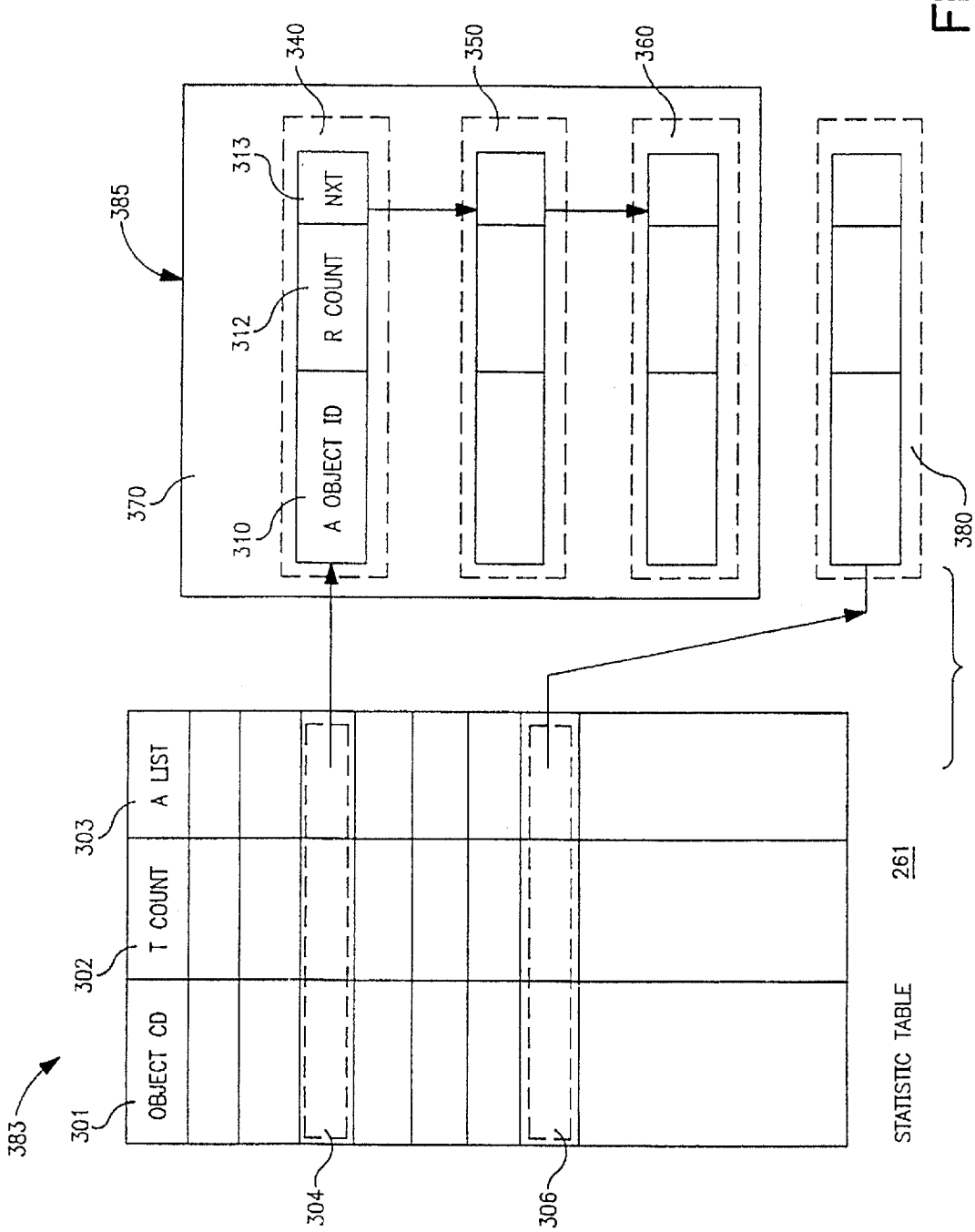

383
385
370
OBJECT CD
T COUNT
A LIST
301
302
303
304
306
261
STATISTIC TABLE
A OBJECT ID
R COUNT
NXT
310
312
313
340
350
360
380

FIG. 3B

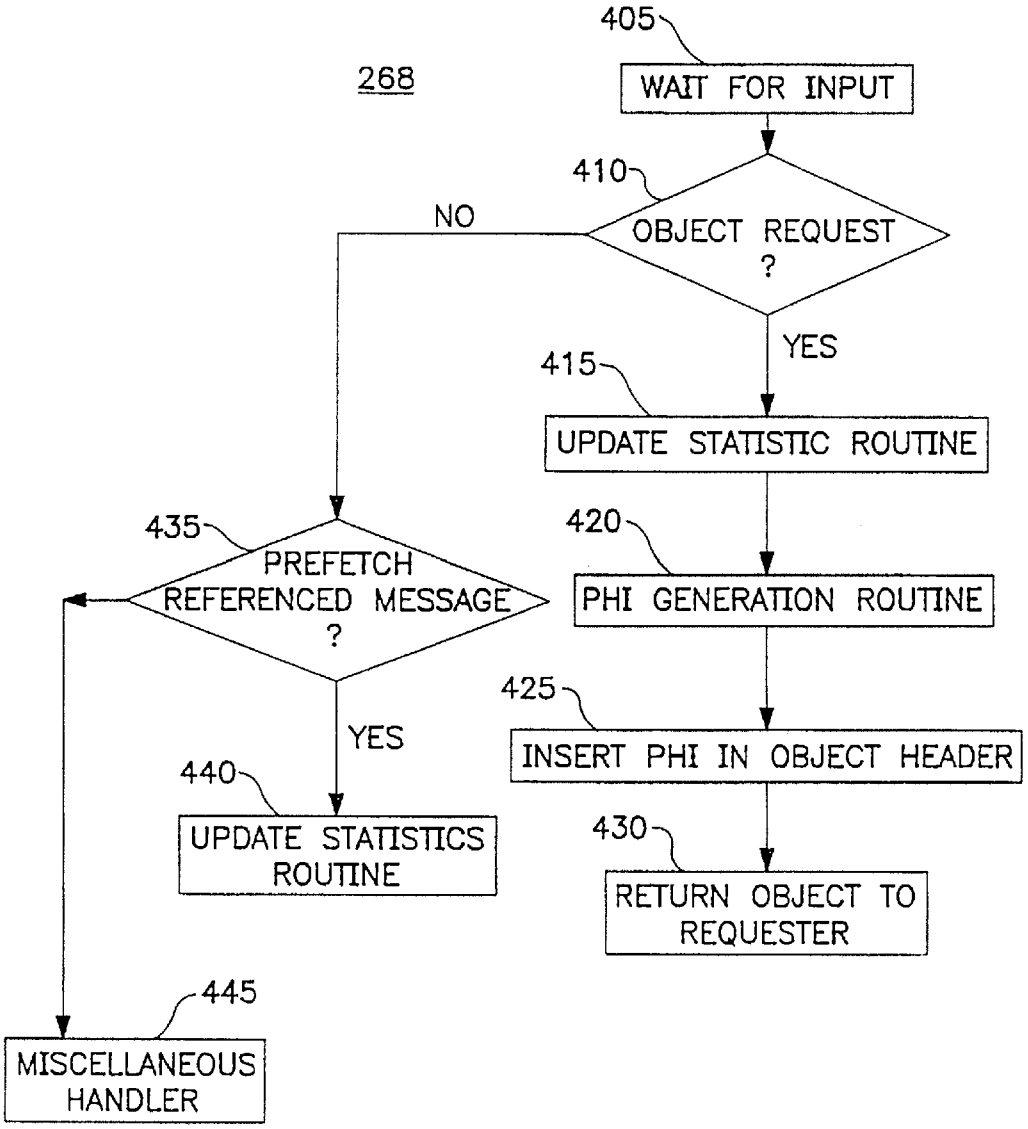

268
405
WAIT FOR INPUT
410
OBJECT REQUEST ?
NO
YES
415
UPDATE STATISTIC ROUTINE
420
PHI GENERATION ROUTINE
425
INSERT PHI IN OBJECT HEADER
430
RETURN OBJECT TO REQUESTER
435
PREFETCH REFERENCED MESSAGE ?
YES
440
UPDATE STATISTICS ROUTINE
445
MISCELLANEOUS HANDLER

FIG. 4

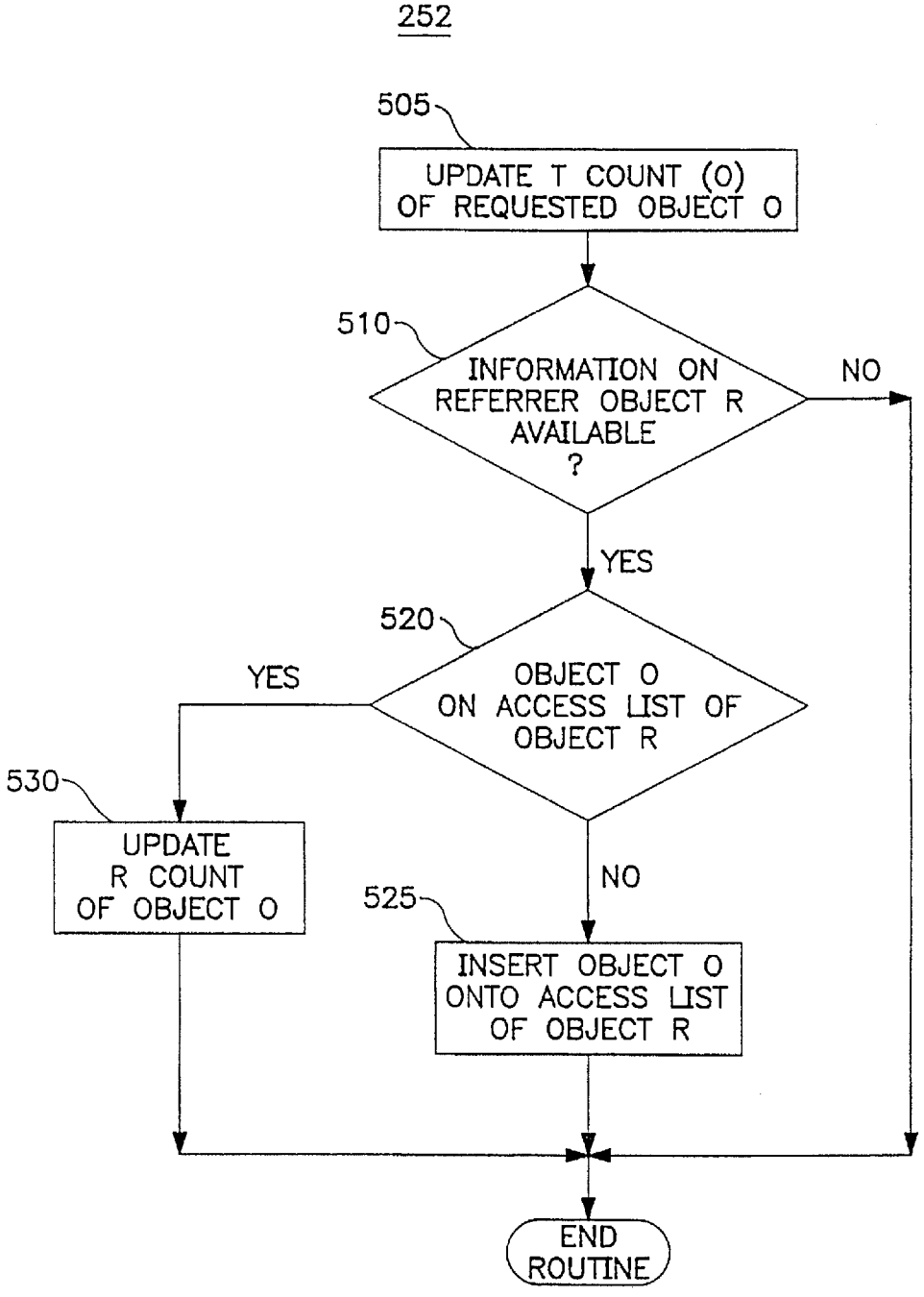

252
505
UPDATE T COUNT (O) OF REQUESTED OBJECT O
510
INFORMATION ON REFERRER OBJECT R AVAILABLE ?
NO
YES
520
OBJECT O ON ACCESS LIST OF OBJECT R
YES
530
UPDATE R COUNT OF OBJECT O
NO
525
INSERT OBJECT O ONTO ACCESS LIST OF OBJECT R
END ROUTINE

FIG. 5

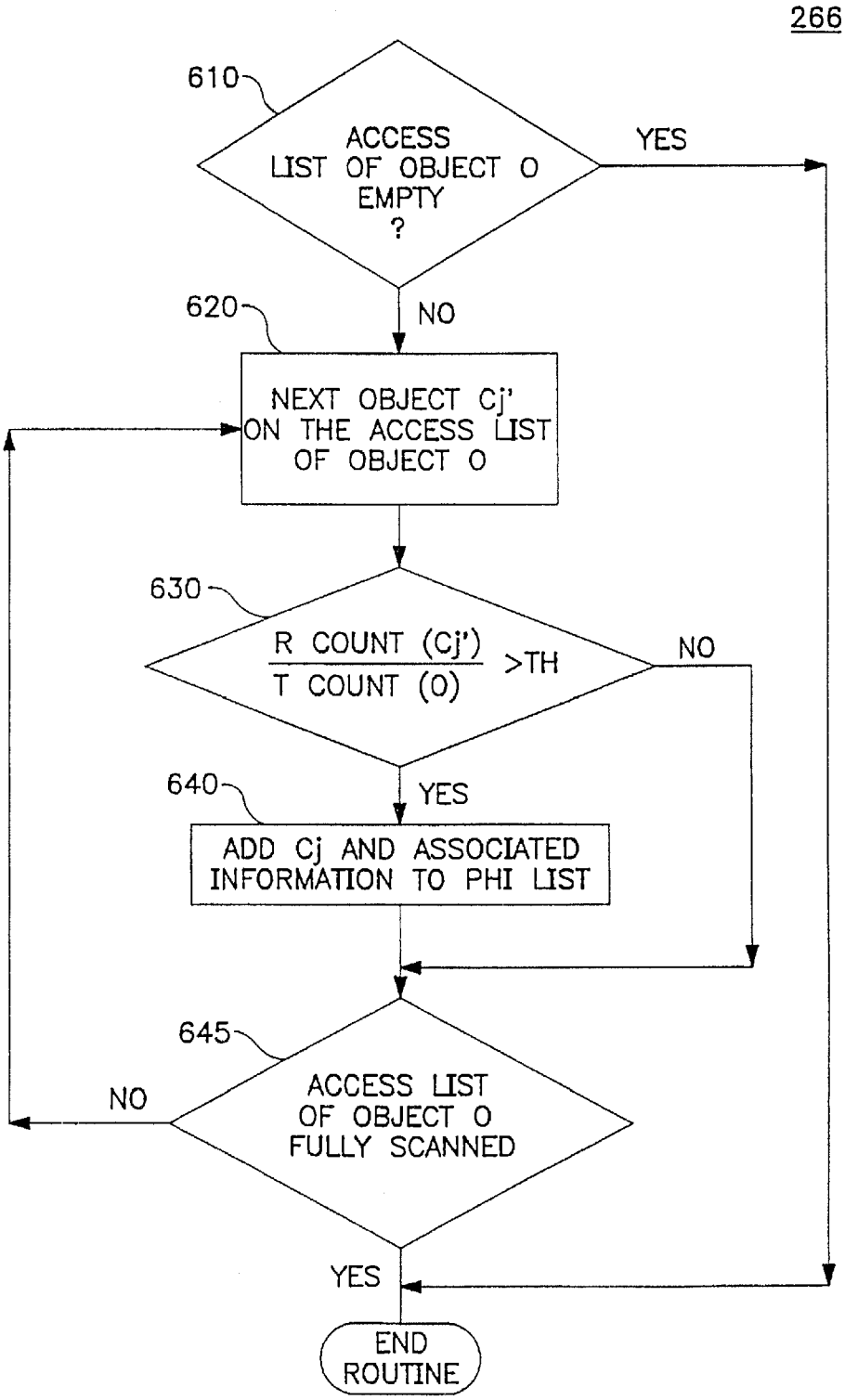

266
610
ACCESS LIST OF OBJECT O EMPTY ?
YES
NO
620
NEXT OBJECT Cj' ON THE ACCESS LIST OF OBJECT O
630
R COUNT (Cj') / T COUNT (O) >TH
NO
YES
640
ADD Cj AND ASSOCIATED INFORMATION TO PHI LIST
645
ACCESS LIST OF OBJECT O FULLY SCANNED
NO
YES
END ROUTINE

FIG. 6

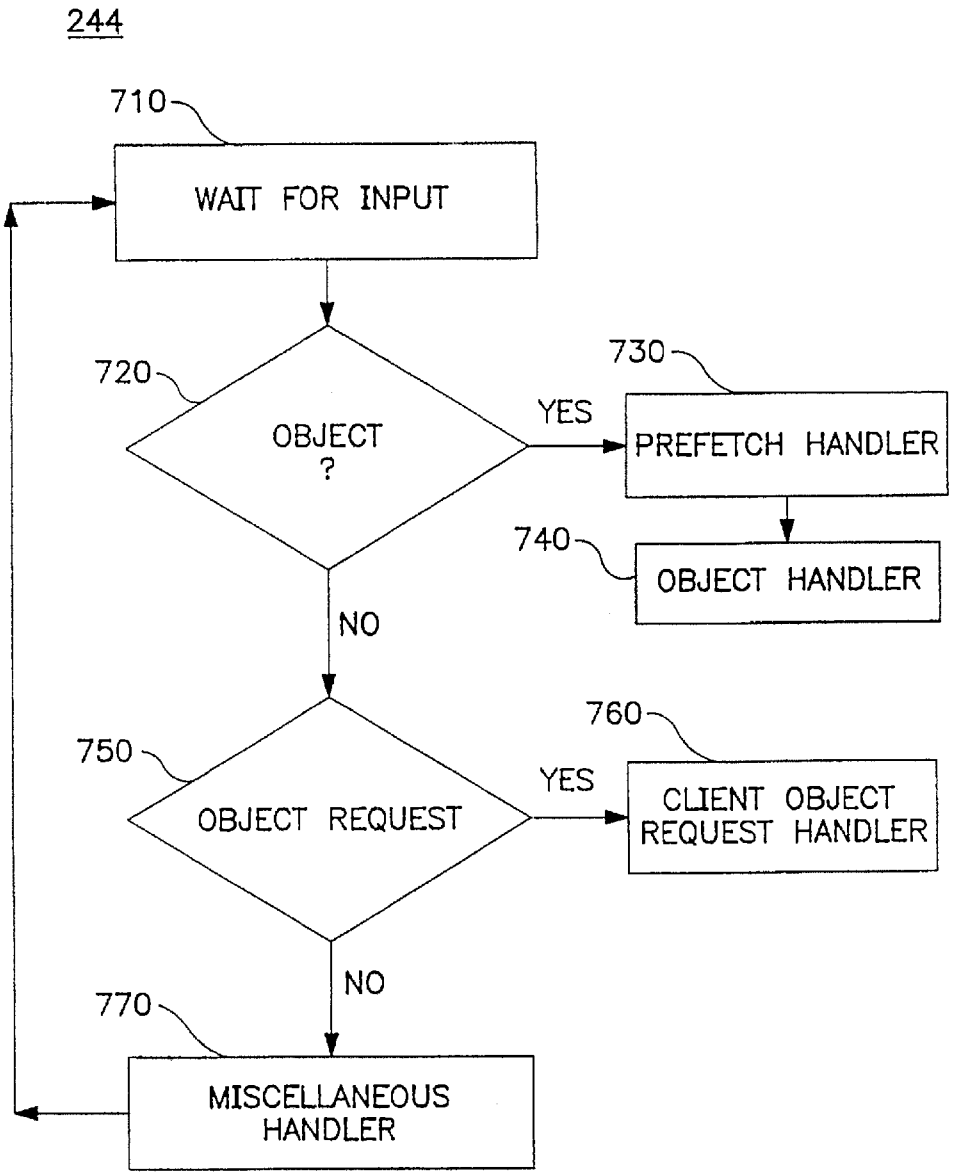

244
710
WAIT FOR INPUT
720
OBJECT ?
YES
730
PREFETCH HANDLER
740
OBJECT HANDLER
NO
750
OBJECT REQUEST
YES
760
CLIENT OBJECT REQUEST HANDLER
NO
770
MISCELLANEOUS HANDLER

FIG. 7

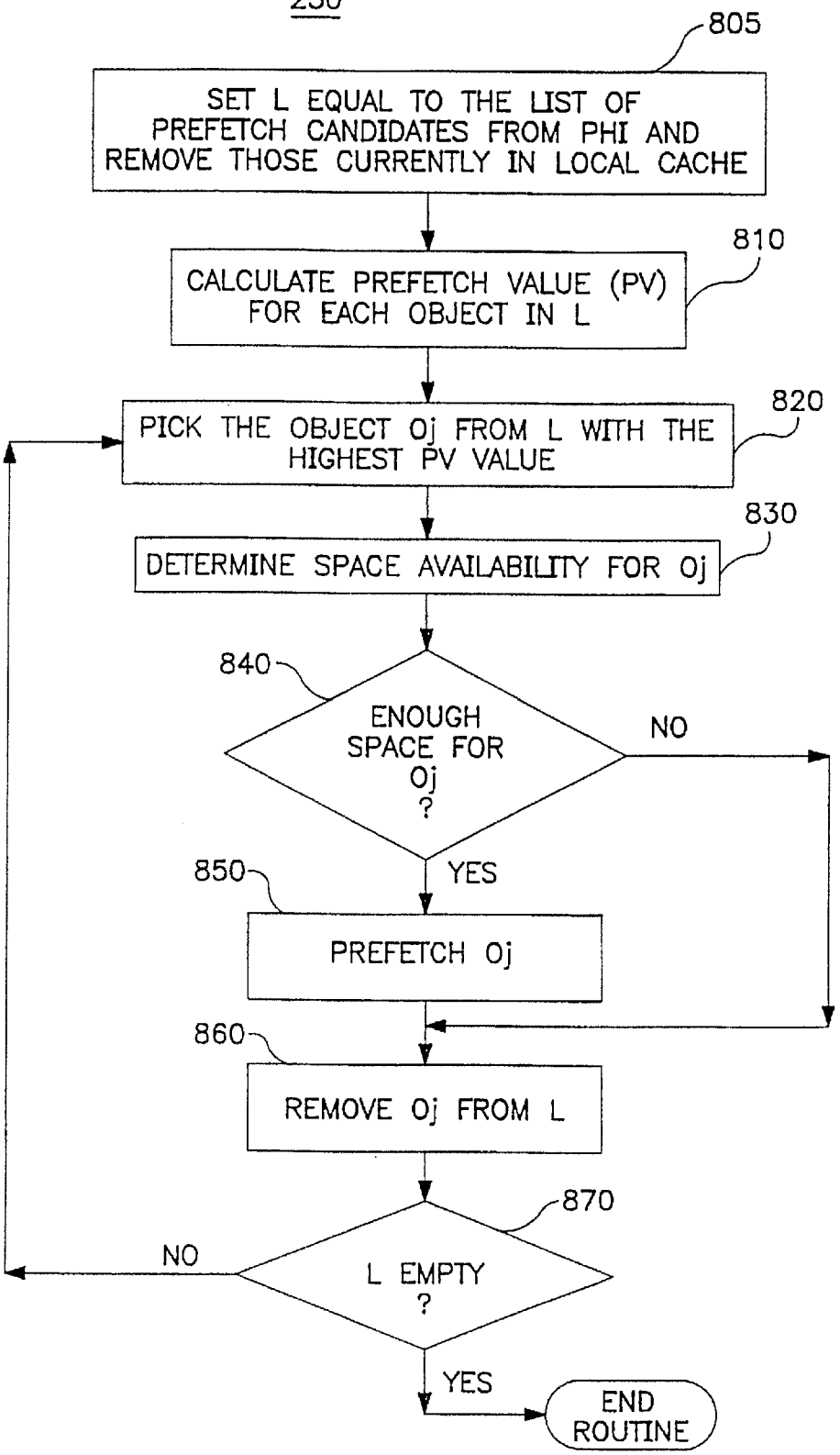

250
805
SET L EQUAL TO THE LIST OF PREFETCH CANDIDATES FROM PHI AND REMOVE THOSE CURRENTLY IN LOCAL CACHE
810
CALCULATE PREFETCH VALUE (PV) FOR EACH OBJECT IN L
820
PICK THE OBJECT Oj FROM L WITH THE HIGHEST PV VALUE
830
DETERMINE SPACE AVAILABILITY FOR Oj
840
ENOUGH SPACE FOR Oj ?
NO
YES
850
PREFETCH Oj
860
REMOVE Oj FROM L
870
L EMPTY ?
NO
YES
END ROUTINE

FIG. 8

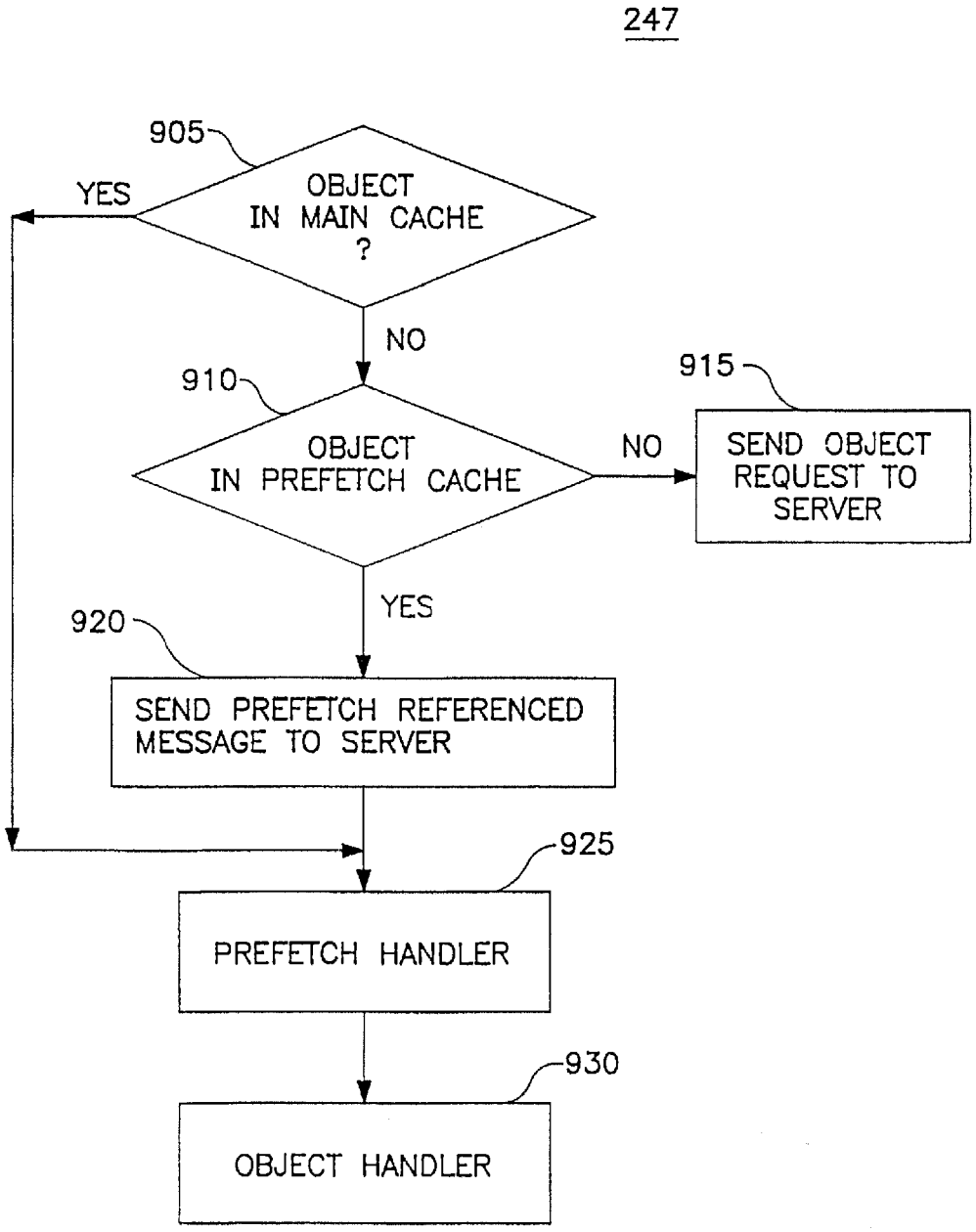

247
905
OBJECT IN MAIN CACHE ?
YES
NO
910
OBJECT IN PREFETCH CACHE
NO
915
SEND OBJECT REQUEST TO SERVER
YES
920
SEND PREFETCH REFERENCED MESSAGE TO SERVER
925
PREFETCH HANDLER
930
OBJECT HANDLER

FIG. 9

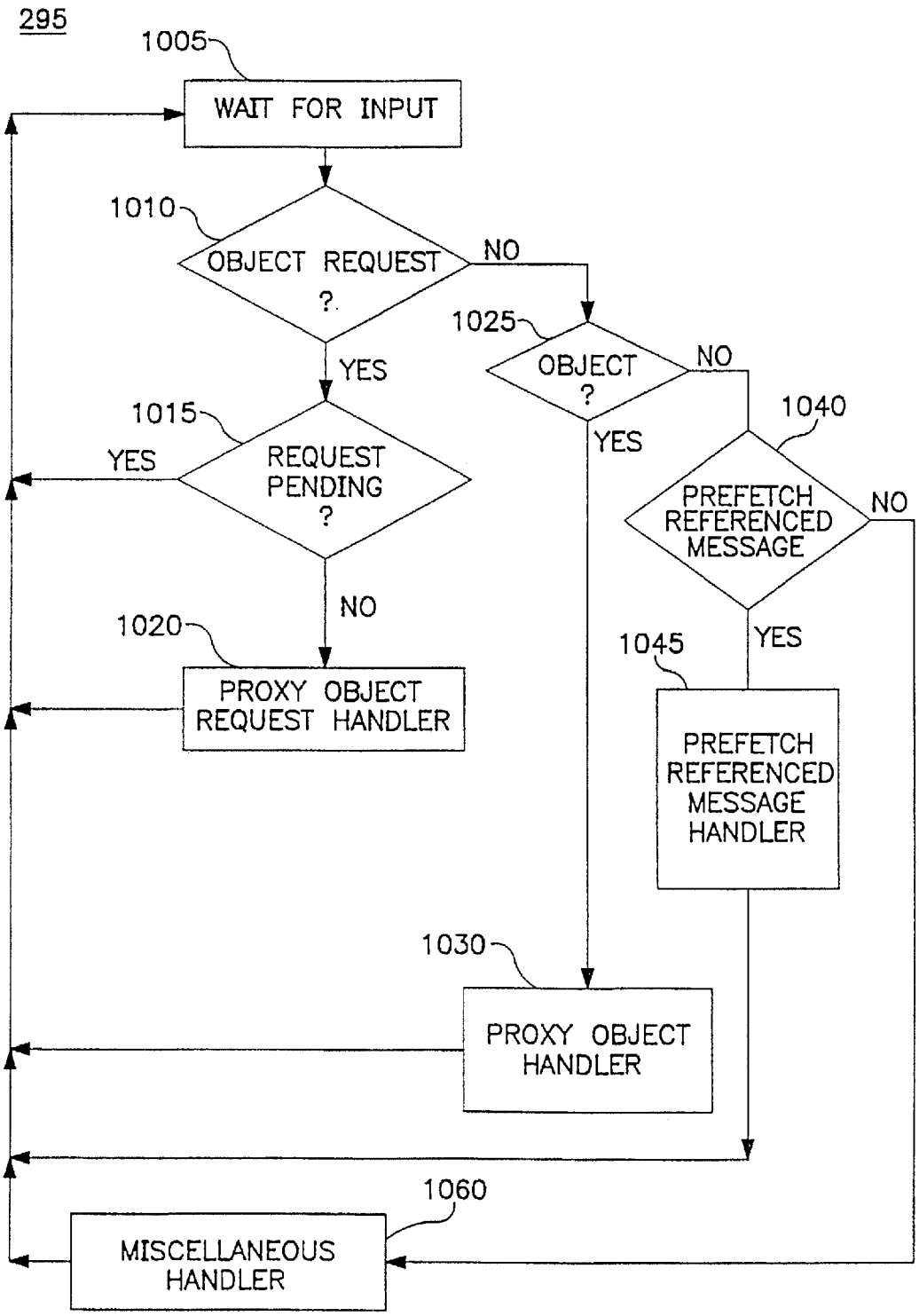

295
1005
WAIT FOR INPUT
1010
OBJECT REQUEST ?
NO
YES
1025
OBJECT ?
NO
YES
1015
REQUEST PENDING ?
YES
NO
1040
PREFETCH REFERENCED MESSAGE
NO
YES
1020
PROXY OBJECT REQUEST HANDLER
1045
PREFETCH REFERENCED MESSAGE HANDLER
1030
PROXY OBJECT HANDLER
1060
MISCELLANEOUS HANDLER

FIG. 10

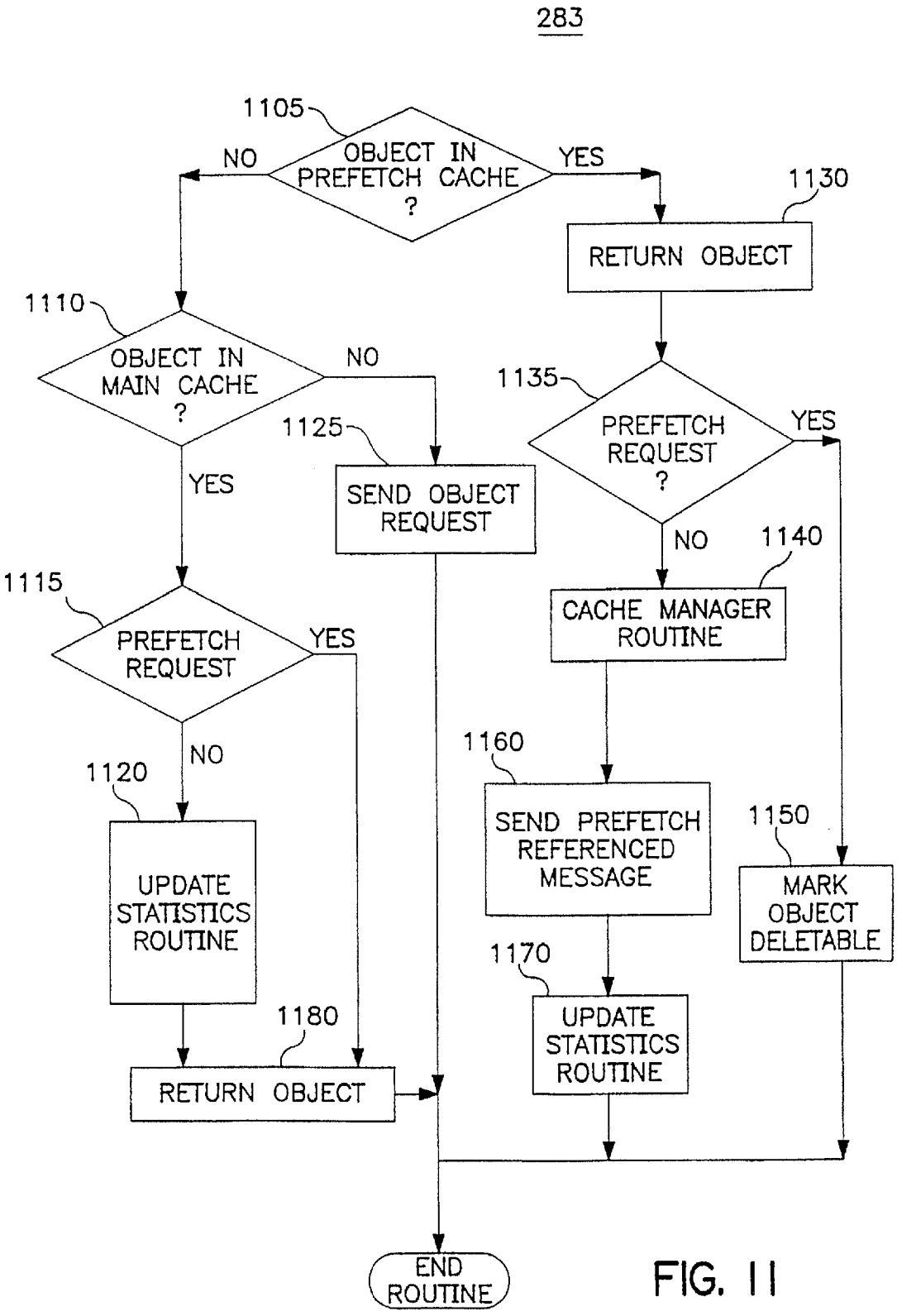

283
1105
OBJECT IN PREFETCH CACHE ?
NO
YES
1130
RETURN OBJECT
1110
OBJECT IN MAIN CACHE ?
NO
1135
PREFETCH REQUEST ?
YES
1125
SEND OBJECT REQUEST
YES
NO
1140
CACHE MANAGER ROUTINE
1115
PREFETCH REQUEST
YES
NO
1160
SEND PREFETCH REFERENCED MESSAGE
1150
MARK OBJECT DELETABLE
1120
UPDATE STATISTICS ROUTINE
1170
UPDATE STATISTICS ROUTINE
1180
RETURN OBJECT
END ROUTINE

FIG. 11

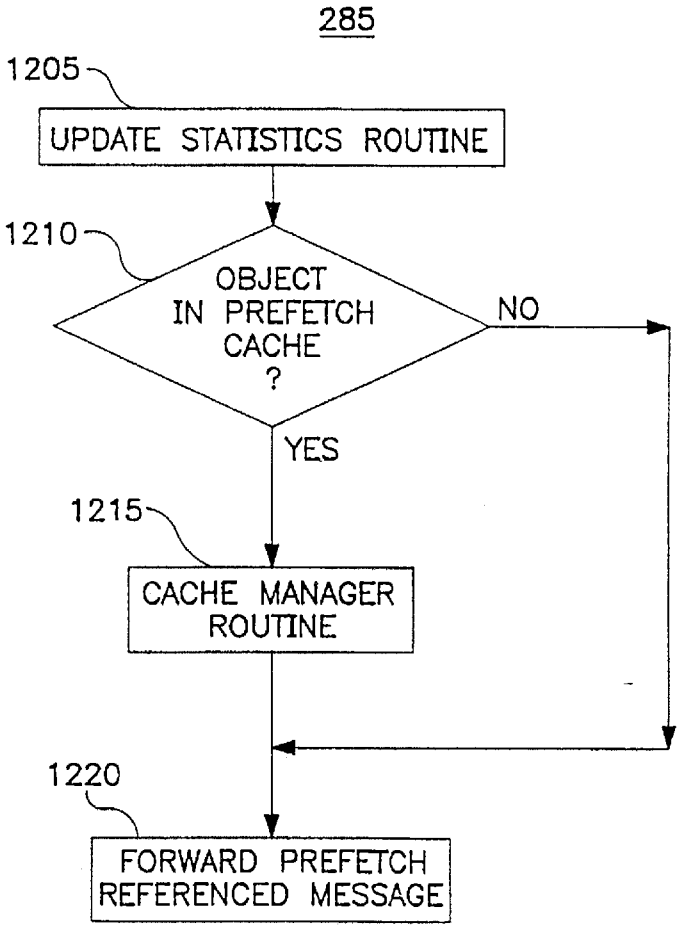

285
1205
UPDATE STATISTICS ROUTINE
1210
OBJECT IN PREFETCH CACHE ?
NO
YES
1215
CACHE MANAGER ROUTINE
1220
FORWARD PREFETCH REFERENCED MESSAGE

FIG. 12

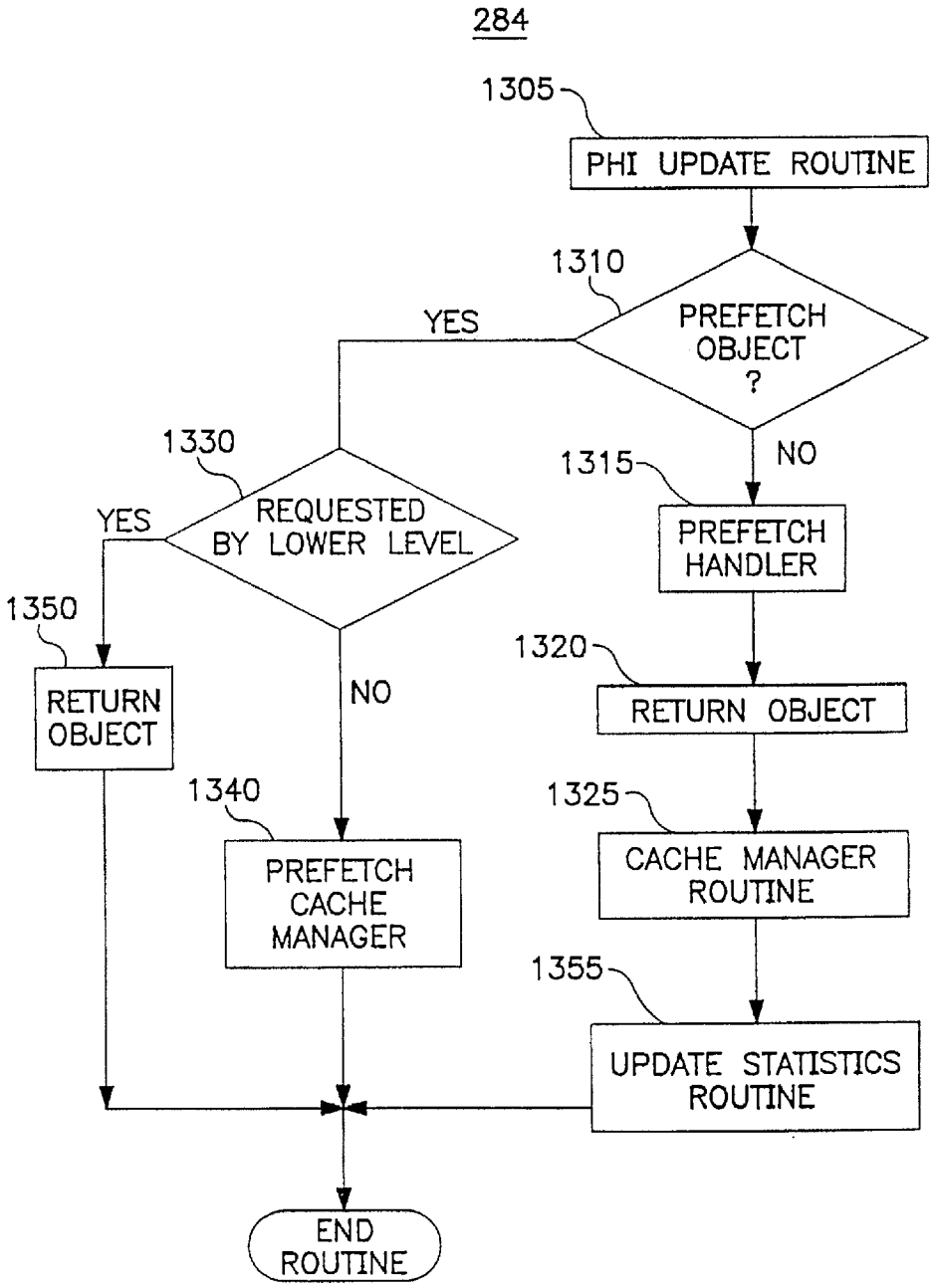

284
1305
PHI UPDATE ROUTINE
1310
PREFETCH OBJECT ?
YES
NO
1330
REQUESTED BY LOWER LEVEL
YES
NO
1315
PREFETCH HANDLER
1350
RETURN OBJECT
1320
RETURN OBJECT
1340
PREFETCH CACHE MANAGER
1325
CACHE MANAGER ROUTINE
1355
UPDATE STATISTICS ROUTINE
END ROUTINE

FIG. 13

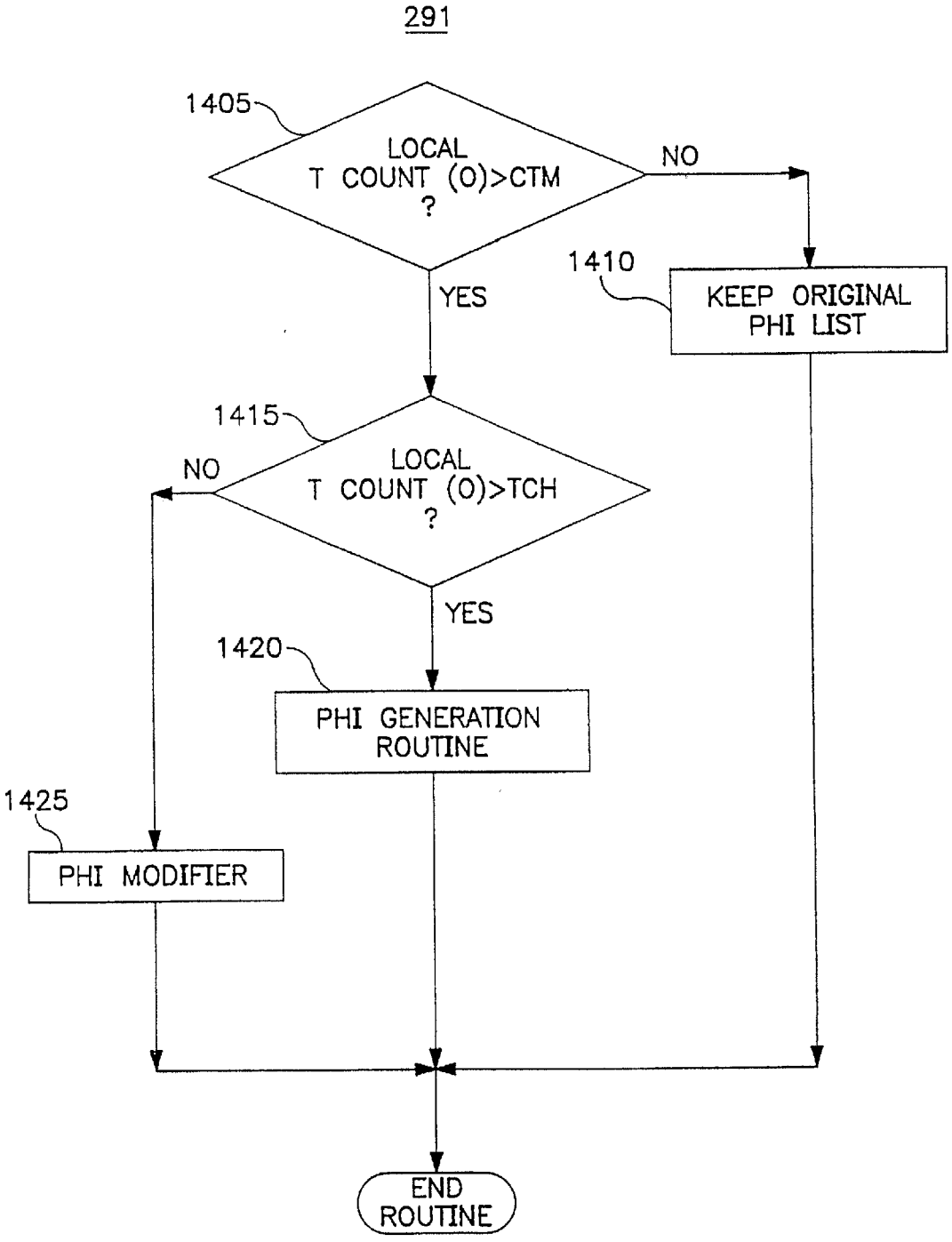

291
1405
LOCAL T COUNT (O)>CTM ?
NO
YES
1410
KEEP ORIGINAL PHI LIST
1415
LOCAL T COUNT (O)>TCH ?
NO
YES
1420
PHI GENERATION ROUTINE
1425
PHI MODIFIER
END ROUTINE

FIG. 14

INTERNATIONAL SEARCH REPORT

Internat al Application No
PCT/GB 98/02920

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WANG Z ET AL: "Prefetching in World Wide Web" IEEE GLOBECOM 1996. COMMUNICATIONS: THE KEY TO GLOBAL PROSPERITY. GLOBAL INTERNET'96. CONFERENCE RECORD (CAT. NO.96CH35942), IEEE GLOBECOM 1996. COMMUNICATIONS: THE KEY TO GLOBAL PROSPERITY. GLOBAL INTERNET'96. CONFERENCE RECORD, LONDON, UK, 18-22 NO, pages 28-32, XP002086567 ISBN 0-7803-3336-5, 1996, New York, NY, USA, IEEE, USA see page 28, right-hand column, paragraph 3 - page 29, right-hand column, paragraph 4.1 see page 30, left-hand column, line 44 - page 31, right-hand column, paragraph 7 --- -/--	1-5,7-9, 12,18,26

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
3 December 1998	21/12/1998

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Fournier, C

1

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No
A	SMITH N G: "The UK national Web cache - The state of the art" COMPUTER NETWORKS AND ISDN SYSTEMS, vol. 28, no. 11, May 1996, page 1407-1414 XP004018238 see page 1409, left-hand column, paragraph 2.2 ---	1,13,15, 18,26
A	ZHIMEI JIANG ET AL: "Prefetching links on the WWW" 1997 IEEE INTERNATIONAL CONFERENCE ON COMMUNICATIONS. TOWARDS THE KNOWLEDGE MILLENNIUM. ICC '97. CONFERENCE RECORD (CAT. NO.97CH36067), PROCEEDINGS OF ICC'97 - INTERNATIONAL CONFERENCE ON COMMUNICATIONS, MONTREAL, QUE., CANADA, 8-12 JUNE 1997, pages 483-489 vol.1, XP002086568 ISBN 0-7803-3925-8, 1997, New York, NY, USA, IEEE, USA see page 483, right-hand column, paragraph 2 - page 484, right-hand column, paragraph 3A ---	1-5,7-9, 12,18,26
A	CHINEN K. & YAMAGUCHI S.: "An Interactive Prefetching Proxy Server for Improvement of WWW Latency" PROCEEDINGS OF THE INET'97 CONFERENCE, June 1997, XP002086569 http://www.isoc.org/INET97/proceedings/a1/a1_3.htm see the whole document ---	1-5,7-9, 12,18,26
P,A	US 5 802 292 A (MOGUL JEFFREY CLIFFORD) 1 September 1998 see the whole document -----	1,18,26

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

Intern. al Application No
PCT/GB 98/02920

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 5802292 A	01-09-1998	NONE	

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) **International Patent Classification** 6 :
H04L 12/18, 29/14

A1

(11) **International Publication Number:** **WO 99/48246**

(43) **International Publication Date:** 23 September 1999 (23.09.99)

(21) **International Application Number:** PCT/US99/05731

(22) **International Filing Date:** 16 March 1999 (16.03.99)

(30) **Priority Data:**
09/039,375 16 March 1998 (16.03.98) US

(71) **Applicant:** REUTERS AMERICA INCORPORATED [US/US]; 88 Parkway Drive South, Hauppauge, NY 11788 (US).

(72) **Inventor:** STIRPE, Paul, A.; 3 Kristina Way, Shoreham, NY 11786 (US).

(74) **Agents:** GLEMBOCKI, Christopher, R. et al.; Banner & Witcoff, Ltd., 11th floor, 1001 G Street, N.W., Washington, DC 20001-4597 (US).

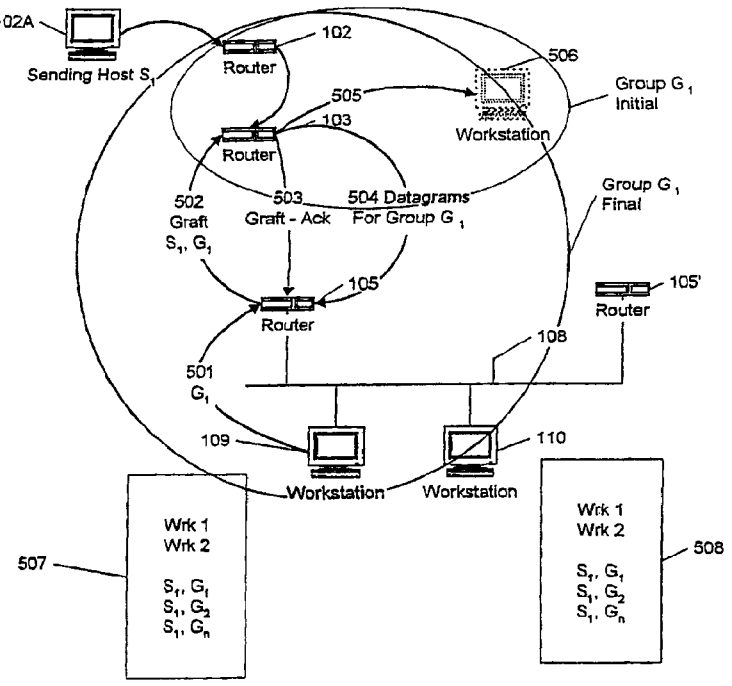
(81) **Designated States:** AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) **Title:** METHOD, APPARATUS, AND MEDIUM FOR MINIMAL TIME MULTICAST GRAFT/JOIN RESTORATION

(57) **Abstract**

The present invention provides a method, apparatus, and medium for quickly re-establishing a lost multicast connection between an end user and a group in a multicast environment. The end user monitors the liveness of the received information as well as retains a list of the multicast communication channels required for re-establishment of a connection to a group. Through the use of an end user-originated Graft or Join based on the stored list of multicast communication channel identifiers (S, G), when the received information is no longer live, an end user may quickly rejoin a multicast group with minimal down time.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

Method, Apparatus, And Medium For Minimal Time Multicast Graft/Join Restoration

Background Of The Invention

1. Technical Field

The invention generally relates to IP multicast technology. More particularly, the invention relates to re-establishing a link with minimal delay, between a user's local area network and a multicast content channel.

2. Related Information

As the Internet becomes increasingly burdened with traffic, solutions for relieving at least some of the burden on the current Internet infrastructure are being sought. Current IP packet transmissions are sent point to point using point-to-point type protocols (PTP) as well known in the art. An example of point-to-point protocol is TCP/IP. Using PTP protocols to send data become increasingly inefficient in terms of bandwidth utilization as final destinations are located closer to each other. Packets traveling to similar destinations may traverse similar paths in a network, thus consuming extraneous bandwidth along the way. Also, the transmitting host faces a mounting burden as it attempts to service the numerous destinations on a timely basis. One solution is the use of IP multicast technology.

Also referred to as IP multicasting, this technology is a mechanism provided by a network that facilitates the transmission of a single packet to a plurality of destinations. When a network node (router or switch) handling the multicast packet in

transit determines that the end destinations to which the packet is heading no longer uses a similar pathway, the network node duplicates and transmits the packet down as many pathways as needed to forward the packet to all predetermined locations specified by the multicast communication channel. The class of controlling protocols utilized by network nodes that build and maintain multicast communication channels (or routing trees) are referred to as IP multicast routing protocols. The class of controlling protocols utilized between destination machines and their first hop network nodes are called Group Management Protocols, such as IGMP. The IGMP specification (identified as Internet Group Management Protocol, Version 2, Network Working Group of the Internet Engineering Task Force, November 1997), as is known in the art, is hereby incorporated by reference.

Multiple IP multicast routing protocols exist, including, but not limited to, PIM dense mode, PIM sparse mode, CBT, and MOSPF. For example, PIM (Protocol Independent Multicast) dense mode is a multicast routing protocol that controls the maintenance of multicast communication channels utilized for transmissions to multicast groups which are densely distributed across a network. Applicable networks include intranets, extranets and the Internet, and other equivalent networks. PIM dense mode uses reverse path multicasting (RPM), also called reverse path forwarding (RPF), to establish and maintain routing trees. RPM is a technique in which a multicast packet (or datagram) is forwarded if the receiving interface on a network node is the one used to forward unicast datagrams to the source of the multicast datagram.

The first step in establishing a routing tree according to the PIM dense mode specification (Protocol Independent Multicast Version 2, Internet Engineering Task Force, April 2, 1997), hereinafter incorporated by reference, is to broadcast the multicast datagram to all PIM DM enabled routers in the network, such that a routing tree is formed that provides a multicast communication channel on which datagrams are carried. This broadcast process is repeatedly performed every three minutes. Figure 1 shows an example of setting up a routing tree in a PIM dense mode IP multicast system. Network 101 includes a plurality of network nodes 102-107. Also, Figure 1 includes LAN 108 as attached to network node 105 and workstations 109 and 110. As is well known in the art, the communication protocol between network node 105 and workstations 109 and 110 on LAN 108 may include the Internet Group Management Protocol, IGMP. In addition, other communication protocols are available. For example, there are several version of IGMP, mainly version numbers 1, 2 and 3, the last of which is a draft proposal in the research community. Other Management Protocols exist. For example, the Cisco Corporation has developed a protocol called CGMP (Cisco Group Management Protocol) which is similar to IGMP, described above. Here, IGMP is used to tell network node 105 (leaf node) that workstations 109 and 110 exist and continue to have interest in the particular multicast communication channel on which datagrams are forwarded to them. In particular, network node 102 receives a datagram from sending host 102A and transmits it to downstream network nodes 103, 104, and 107 as illustrated by transmission paths 111, 112, and 113. Having received the datagram, network nodes

103, 104 and 107 retransmit the datagram to other network nodes to which they are connected, except to the router from which the datagram was received. In this case, network node 103 transmits the datagram to network node 104 via path 114 and to network node 105 via path 115. Network node 104 retransmits the datagram to network node 103 via path 118, network node 105 via path 120, network node 106 via path 121, and network node 107 via path 119. Network node 107 retransmits the datagram to network node 104 via path 116 and network node 106 via path 122. Finally, depending on circumstances, for example, when the datagram was received by network node 106, it may retransmit the datagram to network node 107 via path 117.

The second step in establishing PIM DM routing trees includes pruning back all branches that lead to end stations that have not expressed interest in attaching to the multicast communication channel. Figure 2 shows an example of the pruning process in action in accordance with the PIM dense mode. In Figure 2, it is assumed that either of workstations 109 or 110 have expressed an interest in attaching to the multicast communication channel (that is currently in the broadcast phase, as shown in Figure 1). Router 105 sends prune message 203 to router 104, but does not send a prune message to router 103, because the RPF (the best reverse path to the source of the multicast data) interface on router 105 is the one on which router 103 is connected. Router 105 will not prune itself from the multicast channel in which it is interested. It is assumed, although not shown, that router 106 servers at least one end user workstation that has expressed interest in the multicast channel. As a result, router 106 does not prune its interface towards router 107, because the interface of

router 106 knows that router 107 is router 106's RPF to the sending host 102A. Router 107, however, does send prune message 205 to router 104.

While Figure 2, as relating to PIM dense mode, shows a mechanism used to establish routing trees, other techniques to establish routing trees are known in the art as related to other IP multicasting protocols including protocols PIM sparse mode, CBT, and MOSPF, for example. In particular, PIM sparse mode is another applicable Multicast Routing Protocol (Protocol Independent Multicast-Sparse Mode: Protocol Specification, September 9, 1997), hereinafter incorporated by reference.

The resulting network as created by routing tables stored in the various network nodes is shown in Figure 3A. The resulting multicast channel is shown in Figure 3B. Datagrams, originating from sending host 102A, are transmitted from network node 102 to network nodes 103 and 107 via paths 301 and 302, respectively. Next, the datagrams are transmitted to network nodes 105 and 106, via paths 303 and 305, respectively. For a multicast routing communication channel to terminate at a network node, the network node preferably has at least one destination end user attached to it and that the destination end user has expressed an interest in attaching to the multicast communication channel (via IGMP or some similar Group Management Protocol). It is noted that, while multicast routing communication channels terminate at network nodes, multicast communication channels terminate at end user workstations. As shown in Figure 3A, there are no end users attached to network node 104. Accordingly, while it is possible to establish and maintain a communication path to network node 104 regarding a multicast channel, the lack of

end users connected to network node 104 suggests that the connection for this channel should be pruned back. In an alternative embodiment of the invention, network node 104 may remain connected to receive a multicast channel so as to be available for quick connection to the multicast channel by another end user.

One feature of multicast systems includes the concept of groups. Groups are sets of destinations that participate in a multicast communication channel. The channel generally originates with a single content provider. However, multiple content providers may combine to provide content on the same multicast communication channel. Accordingly, to specify a content channel precisely, one needs to know the source of the information as well as the group identifier to which the channel is directed. Figure 4 shows network node 102 (of network 101, not shown for simplicity) receiving a multicast datagram from sending host 102A and sending the datagram via path 301 to network node 103. Network node 103 transmits the datagram to network nodes 403, 404, and 405. Originally, all the network nodes including 102, 103, 403, 404, 405, and 105 participated in group G_1, during the broadcast phase. After the pruning phase, only network nodes 102, 103, 403, 404, and 405, remain participating in G_1 (105 having pruned back the channel).

Next, if workstation 109 (or 110) which is connected to network node 105, part of network 101, desires to join group G_1 409, it sends a request via IGMP to network node 105 specifying that it wishes to join group G_1. Network node 105 is considered to be a leaf node, in that it is directly connected via a LAN to end users that have expressed interest in the particular multicast channel, and there are no

further routers downstream of router 105 on the LAN. Network node 105 interprets the request from workstation 109 as a command to attempt to join group G_1. Network node 105 sends a Graft message with the payload G_1, S_1 to network node 103 via path 401. In response, network node 103 replies with a Graft acknowledge (Graft-Ack) signal to network node 105 via path 402. When network node 103 receives the next datagram destined for group G_1 409, it transmits the datagram to network node 105 as network node 105 is now part of group G_1 409.

It is an assumption in the above that the state G_1, S_1 is retained indefinitely, in network node 105 from the previous broadcast phase. However, it is possible that the state may not exist in network node 105, in which case the network node 105 cannot send a Graft towards network node 103 because network node 105 does not know from which network node it can obtain the specific multicast channel.

Further, in the above-described system, as well as in other IP multicast systems, if a leaf network node fails, the system may have to wait between 0 and 3 minutes (the PIM dense mode broadcast phase cycle time) or for an average of 1.5 minutes until the multicast system enters the broadcast phase such that state is re-established in network nodes and the channel can re-establish itself. A difficulty here, especially in environments with a need for minimal disconnect time, is that the delay associated with re-establishing a connection to network node 105, if it becomes separated from group G_1, may take an average of 1.5 minutes. The cause of the separation may be due to a variety of reasons including transmission line failure, local router failure, and related problems. This reconnect delay is the average time in which

PIM dense mode enters the re-broadcasting phase as shown in Figure 1. In systems that require faster reconnect times, for example, in financial arenas where the 1.5 minute average delay freezes a trader's information stream so the trader cannot act as desired, the delay associated with waiting until the overall multicast system re-establishes itself is unacceptable. Further, merely decreasing the multicast system's re-broadcast interval substantially increases the amount of non-usable broadcast information sent to all end network nodes, regardless of their interest in the particular communication channel. This approach can be very costly in terms of inefficient use of network bandwidth. The result of this delay may be one reason that systems requiring a fast re-establishment time (high availability requirements) have not readily embraced multicast technology as a solution for efficient information transfer.

In networks with only one leaf router serving a LAN, if that leaf router fails, connectivity to the group is lost. For example, if network node 105 becomes disabled, LAN 108 is prevented from connecting to group G_1. One possible solution is to utilize a more redundant system, in which there are multiple leaf network nodes. For example, one could add another network node 105' to the LAN 108 to the network, by connecting to network node 103 through path 306 as shown in Figure 3A. In the above-described environment, one of these network nodes 105 and 105' (for example, 105') will have been pruned out of the multicast channel during the pruning phase of the PIM DM protocol, resulting in the multicast channel as shown in Figure 3B.

The dotted line to network node 105' in Figure 3A indicates that while 105 and 105' are considered separate network nodes, as part of the PIM DM protocol, one of them (i.e., 105') has pruned itself from the multicast channel 101, leaving the remaining network node (105) as the "forwarder" as represented in Figure 3B. The process to elect which network node is to be the forwarder to LAN 108 is called Assertion. Routers 105 and 105' execute the Assert process, as specified in the PIM dense mode specification, and only one of the network nodes becomes the forwarder, and the other network node(s) on the LAN that lose the assertion process prune themselves back, e.g. 105'. Accordingly, when network node 105' has pruned itself from the multicast channel, it needs to wait for the re-broadcast phase (see Figure 1 implemented every three minutes) before network node 105' can begin receiving new datagrams again (see Figure 3A). When this occurs, the assertion process is repeated and the network nodes that lose the assertion process prune themselves back.

The existence of two client side network nodes (leaf nodes in this case) 105 and 105' is to make LAN 108 more robust in being able to handle router or switch failures (network node). However, at most only one of the two network nodes 105 and 105' is active in receiving datagrams from network node 103 and forwarding them onto the LAN 108, at any given time, because the other node (e.g., 105') was pruned out of the multicast channel during the pruning phase. Note that during the assertion process, both network nodes 105 and 105' can be acting as forwarder to LAN 108, until one of the network nodes wins the election process and becomes the forwarder for the LAN 108. If network node 105 is the forwarded for the LAN 108,

and in the event that network node 105 becomes disabled, workstations 109 and/or 110 on LAN 108 will have to wait until the PIM DM re-broadcast phase initiates, so as to forward packets to network node 105' (an average of 1.5 minutes). When 105' begins to forward packets onto LAN 108, network node 105' will elect itself as the forwarder (since no other network node has challenged it and the assert election process is not triggered in this case) and being to forward packets to onto the LAN 108. While waiting for reconnection to the multicast channel, the hosts on LAN 108 may experience large gaps in messages, and general data loss. Again, the maximum waiting interval is, as above, 3 minutes, with an average waiting interval of 1.5 minutes. In applications that require high availability, with or without high throughput characteristics, the average waiting interval of 1.5 minutes is unacceptable as volumes of data may back up, resulting in excessive data loss, memory usage and delay in attempting to re-establish the data to the hosts on LAN 108. Furthermore, users of real-time applications that require up-to-the-second information cannot tolerate even modest periods of disconnection from the service, e.g. instrument traders using a real-time financial application.

Summary Of The Invention

The present invention overcomes the above-described problems by providing a fast re-establishment of a lost connection to a multicast group. To quickly reconnect an end user (or an end user's local area network, or LAN), the end user's system stores relevant connection information so that, when needed, it directs a network node to re-establish its connection to one or more multicast groups.

Each LAN end user on network 108 may elect a director for the LAN 108 that is responsible for coordinating the recovery of the one or more multicast channels to the LAN. Alternatively, a director may not be elected and each end user may act independently and redundantly perform the operations that the director would handle. Advantages of including a director include limiting the task of initiating re-establishment of the multicast channel to one designated entity, rather than several end user workstations. Advantages of not including a director include redundant processing so that if any portion of the LAN 108 fails, each workstation (109, 110) may be able to re-establish its connection with network 101. To this end, when referencing the director, it will be understood that the director is intended to refer to any workstation performing the functions of the director, including but not limited to any end user's workstation redundantly performing the director's functions where LAN 108 has no director and each end user's workstation acts independently.

The director stores the group information as received from various sources. For example, the director may store this information in a source, group pair (also referred to as an S, G pair). The director also monitors sender "liveness" regarding a monitored group. Liveness, for purposes herein, refers to the monitoring of information received from a source. By monitoring liveness information, the receiver (director or workstation) knows when to expect information from the source. That is, when the source is idle (has no data to send), liveness (heartbeat) messages are sent at predefined intervals or at decaying intervals to inform all end user systems that the channel is still alive. Various sender driven liveness messages are specified in

Holbrook et al.'s "Log-Based Receiver" (H. Holbrook, S. Signhal, D. Cheriton: Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation. Computer Communication Review, Vol. 25, No. 4, Proceedings of the ACM SIGCOMM'95, August 1995) and are incorporated herein by reference.

Once the information has not been received as expected, the director determines that the connection to the source has been lost. Next, the director retrieves the previously stored information (the S, G pair), forms a Graft message, and transmits a Graft message to the all routers address (224.0.0.13) or to a specific leaf router address, if the target leaf router address is well known (configured or learned) by the director. The all routers address is specified in the PIM DM specification. Note that the Graft message could be sent using any defined or reserved address that delivers the message to the LAN network nodes that can process the Graft and re-establish the multicast channel to the LAN. The network node or network nodes receiving the Graft message acts accordingly to re-establish the connection to the group by immediately forwarding the Graft upstream on the Reverse Path Forwarding (RPF) interface, as specified in the PIM DM specification. In this scenario, the director is acting like a network node by forwarding a Graft to its leaf router(s). The receiving network node does not distinguish between the director and another node, in this case, and processes the Graft as if it were received from a network node, as specified in the PIM dense mode specification. The result is that the multicast channel is more rapidly re-established the LAN. Thus, by using the system as embodied by the

invention, a director may quickly re-establish a connection to a group with minimal delay.

As purposes herein, the Internet is used as an example of a network of computers which benefits from IP multicast technology. However, it will be understood that the invention as described herein may be readily applied to internets, extranets, WANs and other networks which may benefit from multicasting. Further, while the invention is described in connection to PIM dense mode, it is readily apparent that the advantages described herein are applicable to other IP multicasting protocols including PIM sparse mode, CBT, and MOSPF, and other equivalent IP multicasting protocols. Accordingly, the reference to PIM dense mode is made by way of example only. For example, in PIM sparse mode, the director would send a Join message rather than a Graft, when the liveness information has not been received as expected. While PIM sparse mode is not a broadcast and prune protocol, as is PIM dense mode, the rapid recovery mechanisms described herein are applicable in achieving rapid multicast channel recovery.

Brief Description Of The Drawings

In the following text and drawings, wherein similar reference numerals denote similar elements throughout the several views thereof, the present invention is explained with reference to illustrative embodiments.

Figure 1 is network diagram showing a broadcast phase of a conventional multicast system.

Figure 2 is network diagram showing a pruning phase of a conventional multicast system.

Figures 3A and 3B are network diagrams showing a remaining network after a broadcast and pruning phases of a conventional multicast system.

Figure 4 is a network diagram showing a Graft in a conventional multicast system.

Figure 5 is a network diagram of a Graft originating with an end user according to embodiments of the present invention.

Figure 6 is a packet format diagram of a multicast system as used in conjunction with embodiments of the present invention.

Figure 7 is a network diagram of the present invention including an embodiment of an end user's system according to embodiments of the present invention.

Figure 8 is a network diagram of the present invention showing Graft messages as contemplated by the present invention.

Detailed Description Of The Preferred Embodiments

The present invention relates to quickly re-establishing a connection to a multicast group. For purposes herein, a multicast group should be understood to also relate to a multicast communication channel.

Figure 5 shows an array of workstations 109 and 110 on network 108 (for example, a LAN) connected via network node 105 to network 101 (not shown for simplicity). While only two workstations are shown, it is understood that many

workstations may be connected to network 108. One of the workstations, 109 or 110 may be elected as director for a particular S,G pair. As the director officiates in numerous occasions, the director may be also referred to as a host of the network 108. In this example, workstation 109 is the elected director. In case director workstation 109 fails, workstation 110 is the designated assistant director. The assistant director stores information similar to workstation 109 so as to replace workstation 109 in the event workstation 109 fails.

To this end, when referencing the director, it will be understood that the director is intended to refer to any workstation performing the functions of the director, including but not limited to any end user's workstation redundantly performing the director's functions where LAN 108 has no director and each end user's workstation acts independently.

Workstation 109 stores information 507 in internal memory. While not shown for simplicity, workstations 109 and 110 may contain various forms of internal memory including RAM, ROM, replaceable storage devices (for example, diskettes, hard drives, and CD-ROMs), and equivalents thereof. Information 507 contains a list of which workstations are the director and assistant director (wrk1 and wrk2 respectively) and a list of S, G pairs that list the source (for example, S_1) of various groups (for example, G_1, G_2, through G_n). Workstation 110 may store a similar set of information 508.

The director derives sender liveness information from information received in connection with the received liveness datagrams, the latter (liveness or heartbeat

mechanisms) being discussed in the Holbrook et al. "Log-Based Receiver". From this sender liveness information, the director determines when to expect datagrams from a source S regarding group G. Once the datagrams or liveness messages are not received as expected, the director transmits a Graft or Join message 501 in PIM DM or PIM SM, respectively, with the appropriate payload (for example, S_n, G_1) to the all routers address (as found in the PIM DM specification). Alternatively, the director transmits the Graft message 501 to a specific leaf router, or any other all local routers address. If the director did not receive the liveness messages as expected, because the forwarder for the multicast channel, e.g. router 105, failed, then the other router, e.g. 105' (or routers) will receive the Graft message and forward it to network node 103 (or on their respective RPF interface), which will cause the quick re-establishment of the multicast channel to LAN 108. Assistant director workstation 110, in addition to being able to perform tasks similar to that of director workstation 109, monitors director workstation 109 in its performance of its director duties. So, if and when director workstation 109 fails, workstation 110 may act as a backup. As noted above, if the PIM sparse mode multicast routing protocol was used instead of PIM dense mode, then the director would send a Join message instead of a Graft. For purposes of simplicity, a workstation and equivalent systems may be referred to as information handlers.

The director generates the S,G pair(s) by monitoring received datagrams for the respective multicast channel(s). The header information in a received IP Multicast packet contains the sender's identity (here the identify of the source end user's

workstation, and the group address G, carried in the IP destination field of the IP header. The workstation stores the sender's information in conjunction with the group over which the datagram was received. For example, in a UDP header, the sender transmits the following: a source address, a destination address, the source port, and the destination port. In a multicast environment, the source address is the IP address of the host sending the information out to receivers. For purposes herein, this source address is the S of the S, G pair. The destination address is the IP address of the multicast communication channel referred to as a group. This group is the group G of the S, G pair. Also, end user workstations can further refine that information that is passed to the end user application for a given G, by filtering those datagrams that are received on destination port numbers of which are interest to the receiving application(s). Workstations 109 and 110 would derive the S, G pair from datagrams received from an ongoing multicast channel via network node 105.

When workstation 109 desires to connect to a multicast channel G_1 (including routers 102 and 103 and workstations 506), it requests connection to this channel G_1 via IGMP as described above. At this point, network node 105 determines which sources (S_1, S_2, etc.) correspond to the requested channel. Next, network node 105 forwards all channels to LAN 108 which correspond to the specified G. So, if S_1,G_1 and S_2,G_1 multicast channel exists in the network, then network node 105 forwards both S_1,G_1 and S_2,G_1 to workstation 109. The application layer in workstation 109 is then responsible for sorting out which, if any, of S_1,G_1 and S_2,G_1 is the actual channel desired. It is possible that the application may desire to obtain all of the S, G pairs.

After workstations 109 and 110, join group G_1, group G_1 is understood to include routers 102, 103, and 105 and workstations 506, 109, and 110.

Figure 6 shows an example of PIM packet formats for control messages as sent from workstation 109 and 110. Bits 0-3 relate to the PIM version, bits 4-7 identify the type of control message, bits 8-15 are reserved, and bits 16-31 are checksum. The following list defines PIM message types:

0 = Hello

1 = Register

2= Register - Stop

3 = Join/Prune

4 = Bootstrap

5 = Assert

6 = Graft

7 = Graft - Ack

8 = Candidate-RP-Advertisement

Here, the Graft control message as part of the present invention is initially formatted in workstation 109 or workstation 110 depending on which workstation(s) is attempting to re-establishing the link to group G. The other control messages can be found in the PIM dense mode or PIM spare mode specification, referenced here in, etc.

Figure 7 shows an embodiment where more than one router is connected to LAN 108. In Figure 7, two routers 701 and 702 connect LAN 108 to router 105. The

interaction between routers (network nodes) 701 and 702 and router (network node) 105 and workstations 109 and 110 is as described above. In some applications, workstation 109 may be the only workstation. In this case, LAN 108 may be supporting a single workstation 109 in connection to network 101. To this end, the term LAN may refer to a device which allows workstation to connect to both network nodes 701 and 702 simultaneously.

Another embodiment of the system described herein is shown with respect to Figure 9. Figure 9 shows network node 105, network nodes 701 and 702, and LAN 108 as described with respect to Figure 7. Also included in the arrangement are network nodes 703 and 704 also connected to network node 105 and LAN 705. Next, both LANs 108 and 705 are connected to workstation 109. In this embodiment, both LANs 108 and 705 support a single (or multiple) workstations. By using this arrangement, workstation 109 (and other similarly connected workstations) is provided with multiple paths to connect to network node 105. An example of when the arrangement Figure 9 may be used is in applications where redundant paths to the network 101 is required, so as to achieve higher availability of services provided by sources such as 706.

Figure 8 shows a network level diagram of network 801 as including a variety of sources S_n and a variety of end user workstations 810-812. Source S_1 transmits group G_1 datagrams to receivers S_4 and S_5 (S_4 and S_5 are receivers with respect to group G_1, but are senders with respect to the groups they source) on multicast channel S_1 G_1. Notably, the combination of receivers S_4 and S_5 comprise group G_1.

Also, source S_2 sends content on group G_1 (on multicast channel $S_2 G_1$) as shown by the combined use of G_1 from sources S_1 and S_2. Alternatively, the content of G_1 may originate with S_1 and use S_2 as a backup for content. Source S_1 also provides information to receiver S_3. Having received datagrams for group G_2, receiver S_3 reformulates and retransmits the datagrams to receivers of network 809. Network 809 may be a LAN or individual groups of receivers so long as they maybe collectively referred to as group G_5. Similarly, receiver S_4 reformulates and retransmits received datagrams to network 808 as group G_3. Finally, receiver S_5 reformulates and retransmits received datagrams to network 807 as group G_4.

When receiver 810 (for example, a workstation with the functionality of workstation 109 of Figure 5) from network 809 wishes to re-establish a connection to group G_5, receiver 810 transmits a Graft message as described above to its leaf router containing the S_3,G_5 pair information. The network node (not shown) in between S_3 and LAN 809 receives G (here, G_5) and sends receiver 810 the multicast channel corresponding to S_3,G_5. In one embodiment, the Graft sent by the director specifically only re-establishes the specific S,G pair and not all *,G pairs, as is done with simply a IGMP join message from the host. Alternatively, all *,G pairs are re-established then unwanted ones dropped.

Similar operations from workstations 811 and 812, when they generate to their respective leaf routers Graft messages specifying the S_4, G_3 pair and the S_5, G_4 pair, respectively.

It should be noted that the relationships as described between S and G for Figure 8 are relative to the datagrams originating with sources S_1 and S_2. When a network receives information from alternative sources, for example, network 808 receiving group G_n datagrams (not shown for simplicity) from source S_3, the Graft S, G pair information from receiver 811 takes the form of (S_3, G_n).

It is readily understood that the method described above may be implemented through the use of a computer-readable medium. For example, one may store computer-implemented steps in a medium which is then read by any of workstations 109 and 110. Further, the process steps may be transferred into an internal storage of workstations 109 and 110 (or into alternative network nodes, for instance) through a different input into LAN 108 or network 101.

Using the above-described system, delay times in re-establishing a connection to a group may be minimized through sensing a fault in received (or not received) information, determining how to quickly reconnect, and reconnecting to a group through the use of information stored in a monitoring station.

It will be apparent to those skilled in the art that application of the present invention need not be utilized in conjunction with the Internet. It is also envisioned that the techniques of the present invention apply to any network including intranets and extranets.

Further, the present invention may also be implemented in a peer-to-peer computing environment or in a multi-user host system having a mainframe or a minicomputer. Thus, the computer network in which the invention is implemented

should be broadly construed to include any multicast computer network from which a client can retrieve a channel in a multicast environment.

In the foregoing specification, the present invention has been described with reference to specific exemplary embodiments thereof. In particular, reference has been made to the PIM dense mode specification. Although the invention has been described in terms of a preferred embodiment, those skilled in the art will recognize that various modifications, embodiments or variations of the invention can be practiced within the spirit and scope of the invention as set forth in the appended claims. Some variations include the use of PIM sparse mode, CBT, and MOSPF IP multicast protocols and other multicast protocols. All are considered within the sphere, spirit, and scope of the invention. The specification and drawings are, therefore, to be regarded in an illustrated rather than restrictive sense. Accordingly, it is not intended that the invention be limited except as may be necessary in view of the appended claims.

CLAIMS:

I claim:

1. A system for re-establishing a connection to a multicast channel comprising:

a first network node receiving multicast data from a network;

a second network node connected to said network;

an information handler connected to said first network node and said second network node, said information handler including

a memory for storing reconnect information, and

one or more monitoring device(s) that monitors the reception of said data for a disconnect of said network node from said multicast channel and alerts said information handler upon occurrence or detection of said disconnect,

wherein, upon reception of said alert, said information handler retrieves said reconnect information and transmits said information to at least said second network node so as to reconnect said information handler to said multicast channel.

2. The system according to claim 1, wherein said information handler transmits said information to a reserved address specifying all routers.

3. The system according to claim 1, wherein the transmission of said information is in the form of a well formed packet.

4. The system according to claim 3, wherein the well formed packet is in the form of a PIM dense mode Graft.

5. The system according to claim 3, wherein the well formed packet is in the form of a PIM spare mode Join.

6. The system according to claim 1, wherein said reconnect information includes the source of said data and the group to which said data belong.

7. The system according to claim 1, wherein said information handler is a workstation.

8. The system according to claim 1, wherein said network routing protocol is based on the PIM dense mode specification.

9. The system according to claim 1, wherein said network routing protocol is based on the PIM sparse mode specification.

10. The system according to claim 1, wherein said first network node and said second network node are the same node.

11. A method for re-establishing a connection to a multicast channel comprising the steps of:

receiving in a first network node multicast data from a network;

storing in a memory in an information handler, which is connected to said first network node, reconnect information related to establishing a reconnection to said multicast channel;

monitoring the reception of said data for a disconnect of said network node from said multicast channel;

alerting said information handler upon occurrence or detection of said disconnect;

retrieving said reconnect information by said information handler upon reception of said alert; and,

transmitting said information to at least one of said first network node and a second network node to reconnect at least one of said first network node and said second network node to said multicast channel.

12. The method according to claim 11, wherein said transmitting step transmits said reconnect information to a reserved address specifying all routers.

13. The method according to claim 11, wherein said transmitting step transmits said reconnect information in the form of a well formed packet.

14. The method according to claim 13, wherein the well formed packet is in the form of a PIM dense mode Graft.

15. The system according to claim 13, wherein the well formed packet is in the form of a PIM spare mode Join.

16. The method according to claim 11, wherein said reconnect information includes the source of said data and the group to which said data belong.

17. The method according to claim 11, wherein said information handler is a workstation.

18. The method according to claim 11, wherein said network routing protocol is based on the PIM dense mode specification.

19. The method according to claim 11, wherein said network routing protocol is based on the PIM spare mode specification.

20. A computer-readable medium having computer-executable instructions for performing the steps comprising:

receiving at a first network node multicast data related to a multicast channel from a network;

storing reconnect information in a memory in an information handler, which is connected to said first network node;

monitoring the reception of said data for a disconnect of said first network node from said multicast channel;

alerting said information handler upon occurrence or detection of said disconnect;

retrieving said reconnect information by said information handler upon reception of said alert ; and,

transmitting said reconnect information to at least one of said first network node and a second network node to reconnect at least one of said first network node and said second network node to said multicast channel.

21. The computer-readable medium according to claim 20, wherein said transmitting step transmits said reconnect information to a reserved address specifying all routers.

22. The computer-readable medium according to claim 20, wherein said transmitting step transmits said reconnect information in the form of a well formed packet.

23. The computer-readable medium according to claim 22, wherein the well formed packet is in the form of a PIM dense mode Graft.

24. The computer-readable medium according to claim 22, wherein the well formed packet is in the form of a PIM sparse mode Join.

25. The computer-readable medium according to claim 20, wherein said reconnect information includes the source of said data and the group to which said data belong.

26. The computer-readable medium according to claim 20, wherein said information handler is a network node.

27. The computer-readable medium according to claim 20, wherein said information handler is a workstation.

28. The computer-readable medium according to claim 20, wherein said network is based on the PIM dense mode specification.

29. The computer-readable medium according to claim 20, wherein said network is based on the PIM sparse mode specification.

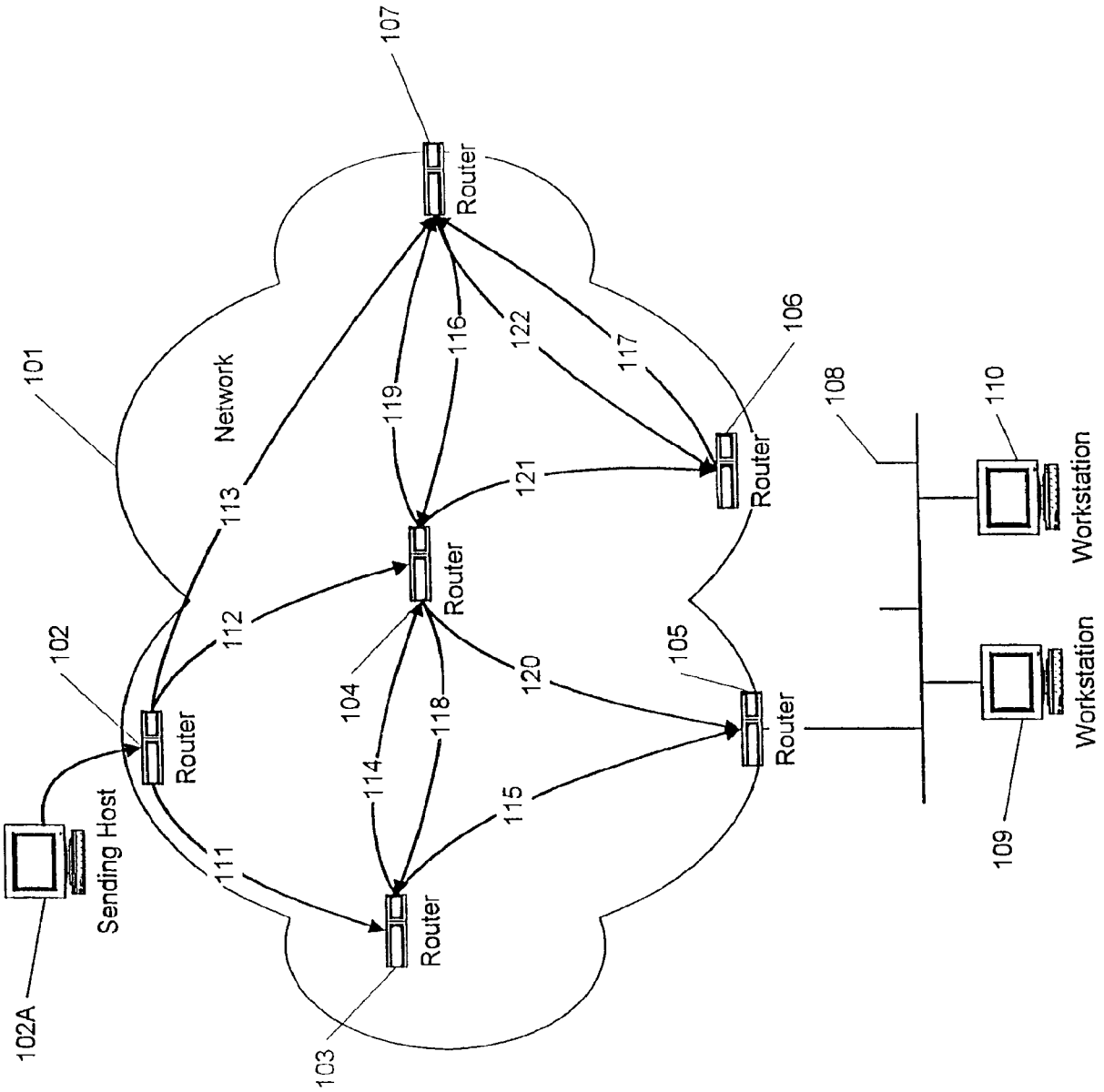

101
Network
102
Router
Sending Host
102A
103
Router
104
105
Router
106
Router
107
Router
108
109
Workstation
110
Workstation
111
112
113
114
115
116
117
118
119
120
121
122

FIG. 1

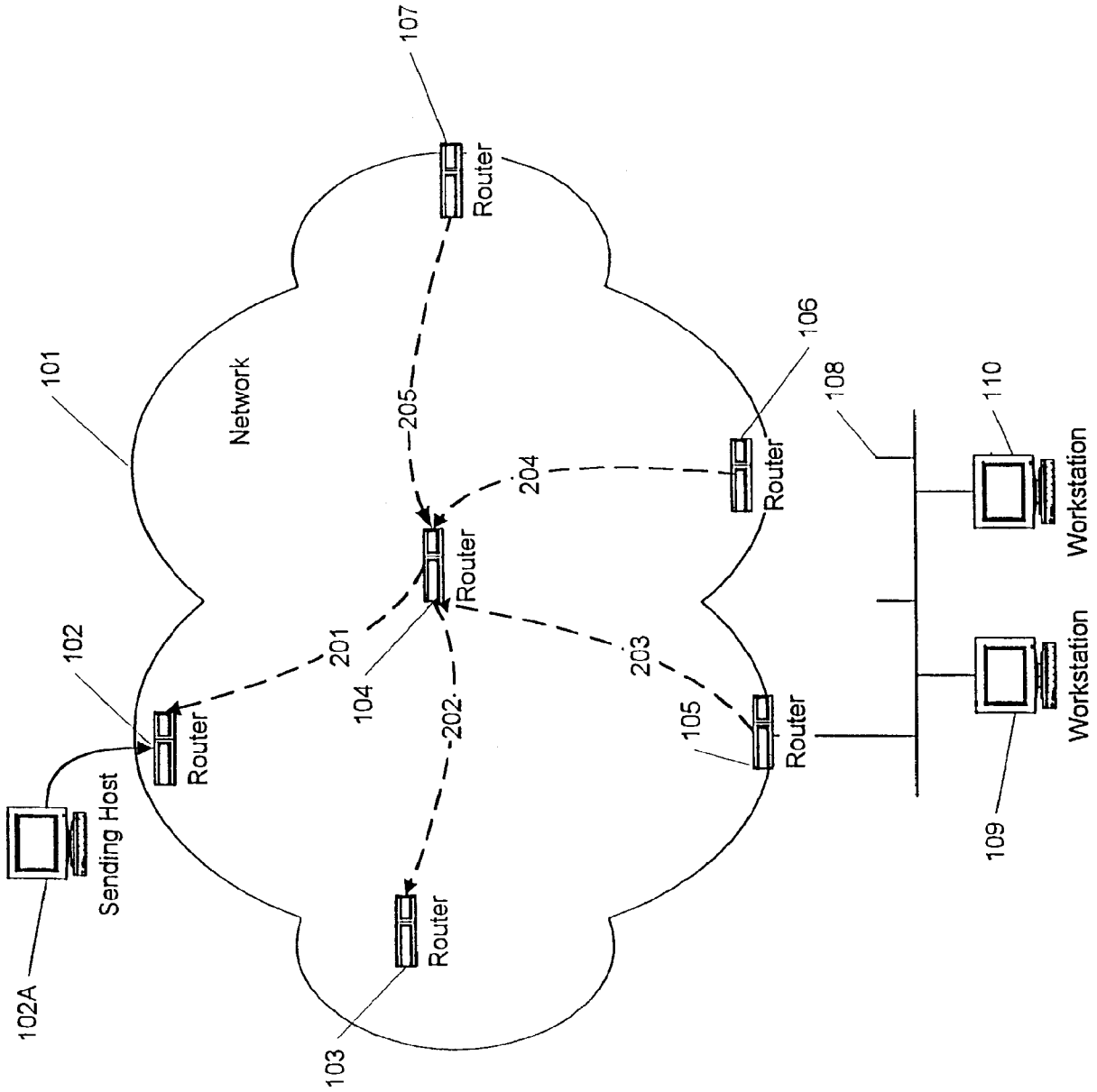

Network
101
Router
107
Router
106
Router
108
110
Workstation
Workstation
109
Router
105
Router
103
Router
102
Sending Host
102A
201
202
203
204
205
104

FIG. 2

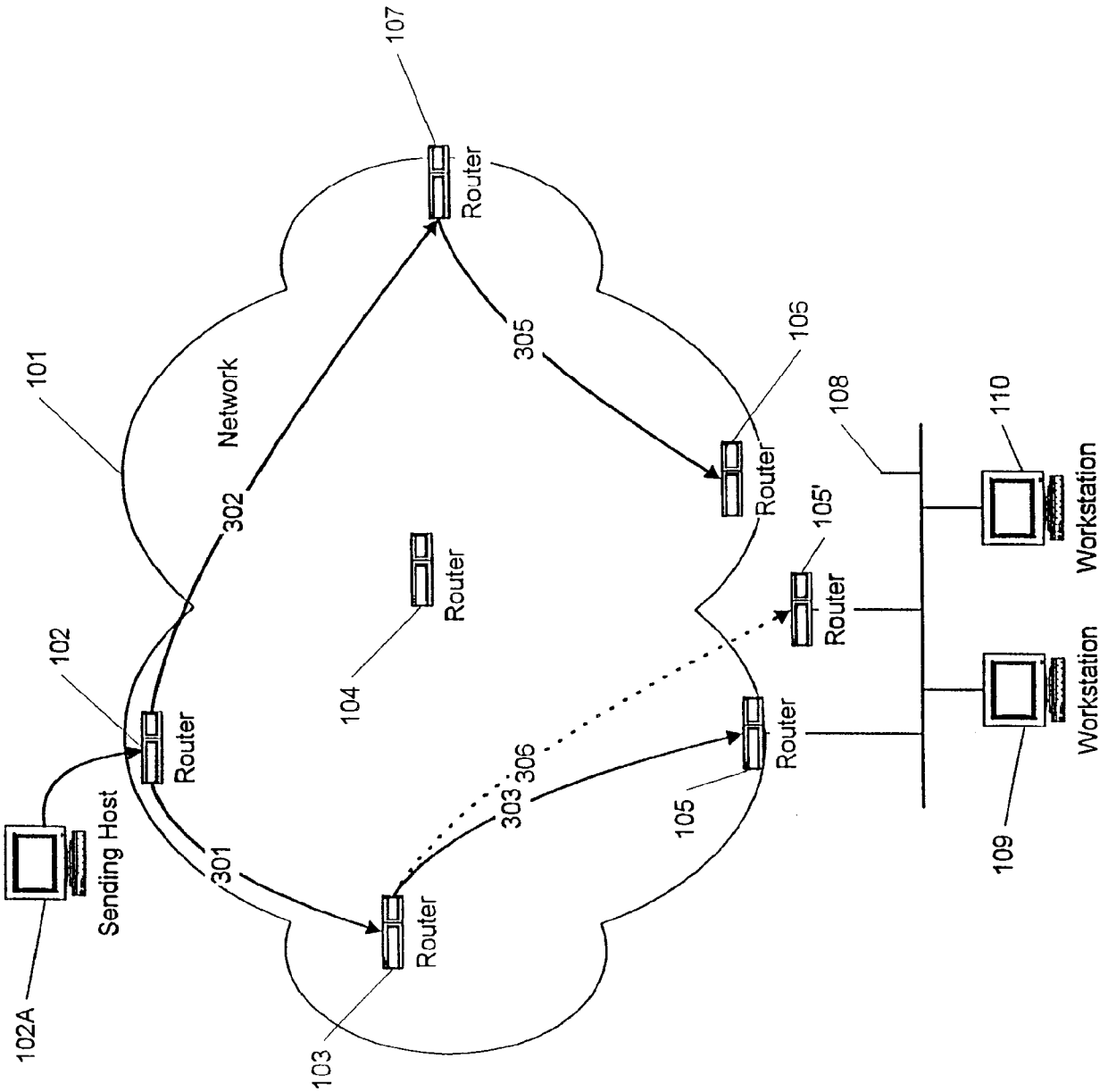

101
Network
102
Router
Sending Host
102A
103
Router
104
Router
105
Router
105'
Router
106
Router
107
Router
108
109
110
Workstation
Workstation
301
302
303
305
306

FIG. 3A

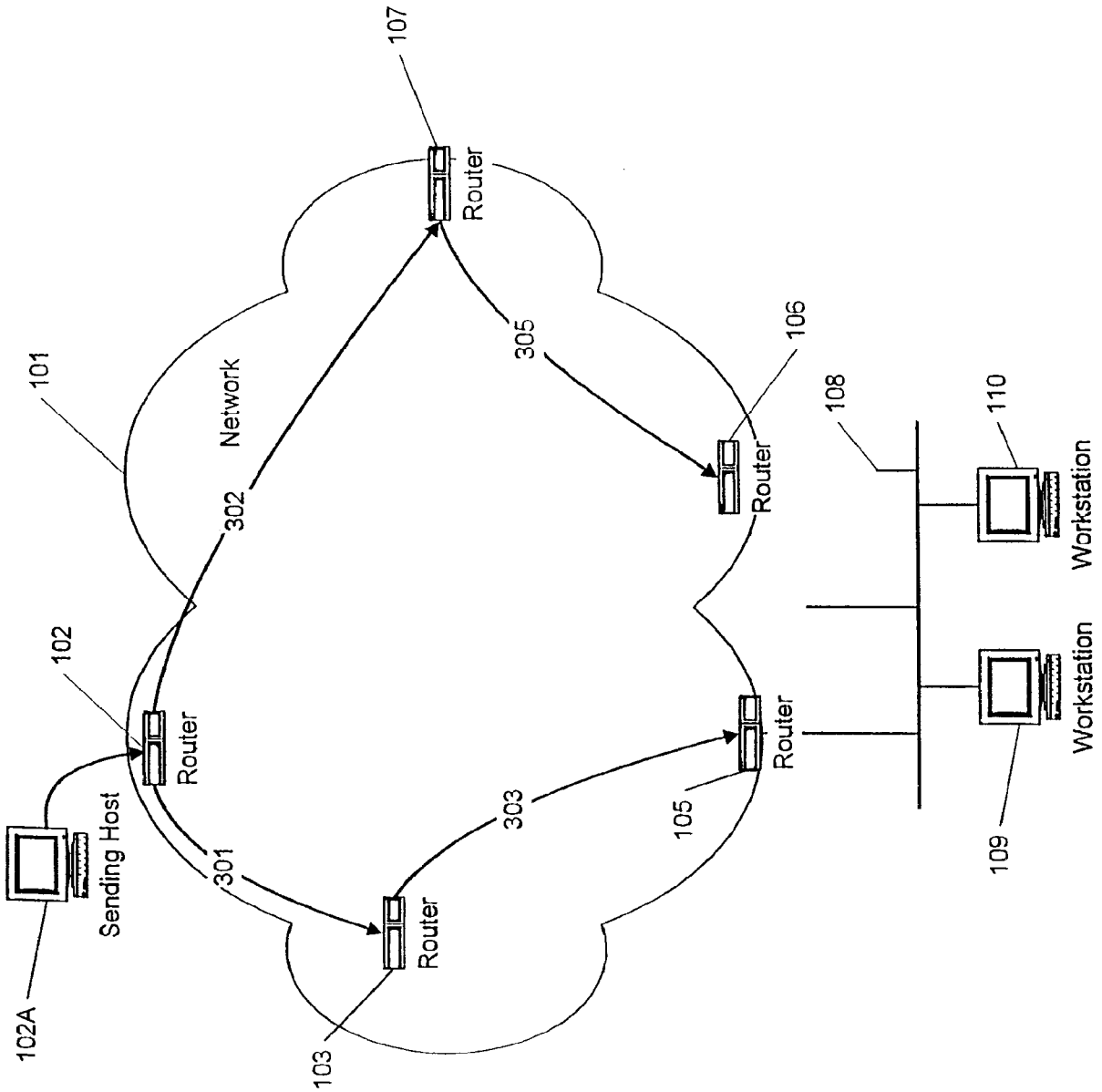

107
Router
101
Network
305
106
Router
108
110
Workstation
302
102
Router
Workstation
Sending Host
303
105
Router
301
Router
109
102A
103

FIG. 3B

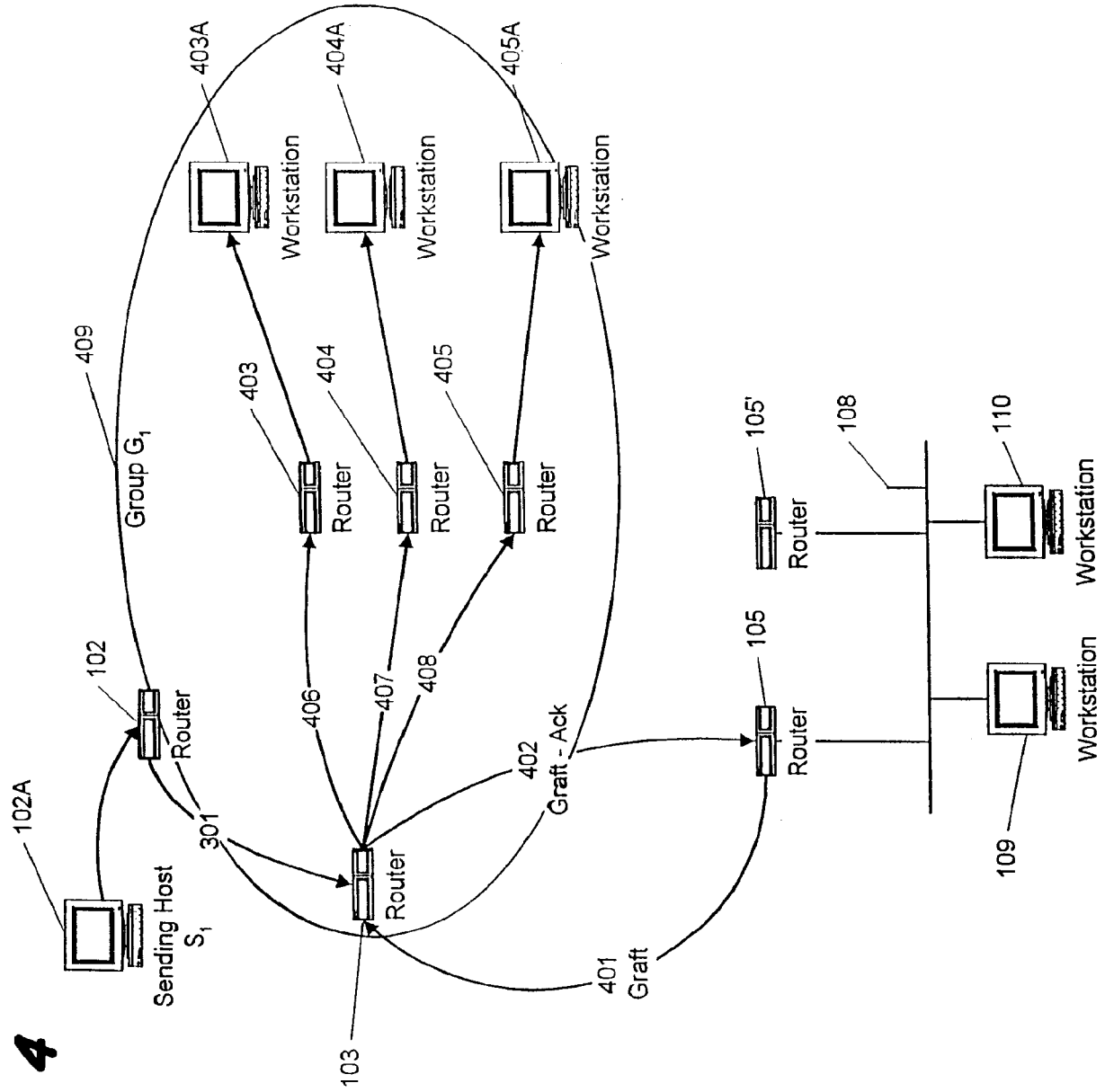

403A
404A
405A
Workstation
Workstation
Workstation
409
Group G_1
403
404
405
Router
Router
Router
105'
108
110
Router
Workstation
102
406
407
408
105
Router
Workstation
102A
301
402
Graft - Ack
Sending Host
S_1
Router
109
103
401
Graft

FIG. 4

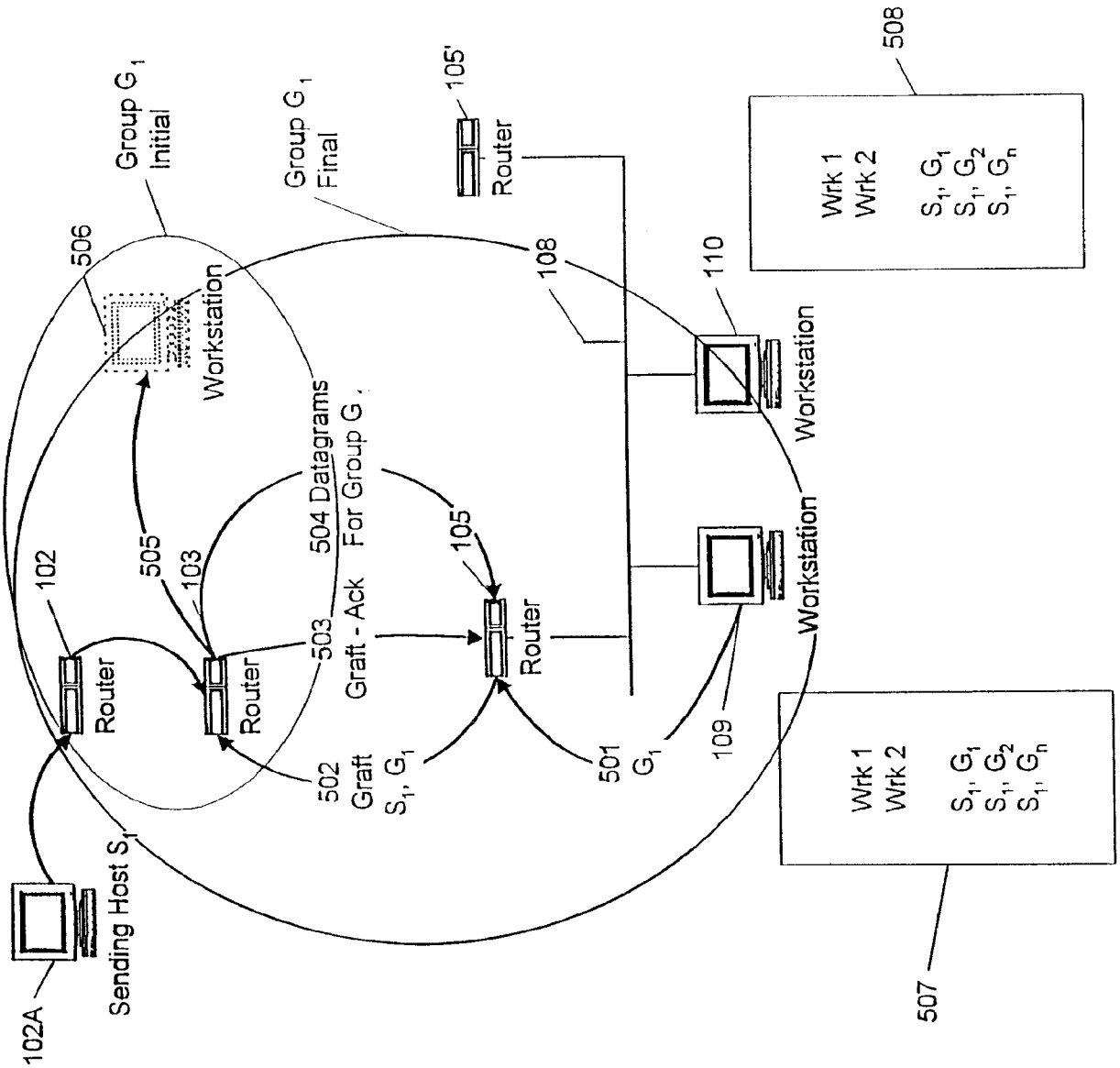

Group G_1 Initial
Group G_1 Final
105'
Router
508
Wrk 1
Wrk 2
S_1, G_1
S_1, G_2
S_1, G_n
506
Workstation
108
110
Workstation
504 Datagrams For Group G
505
103
102
Router
Router
503
Graft - Ack
105
Router
Workstation
502
Graft
S_1, G_1
501
G_1
109
Wrk 1
Wrk 2
S_1, G_1
S_1, G_2
S_1, G_n
Sending Host S_1
102A
507

FIG. 5

FIG. 6

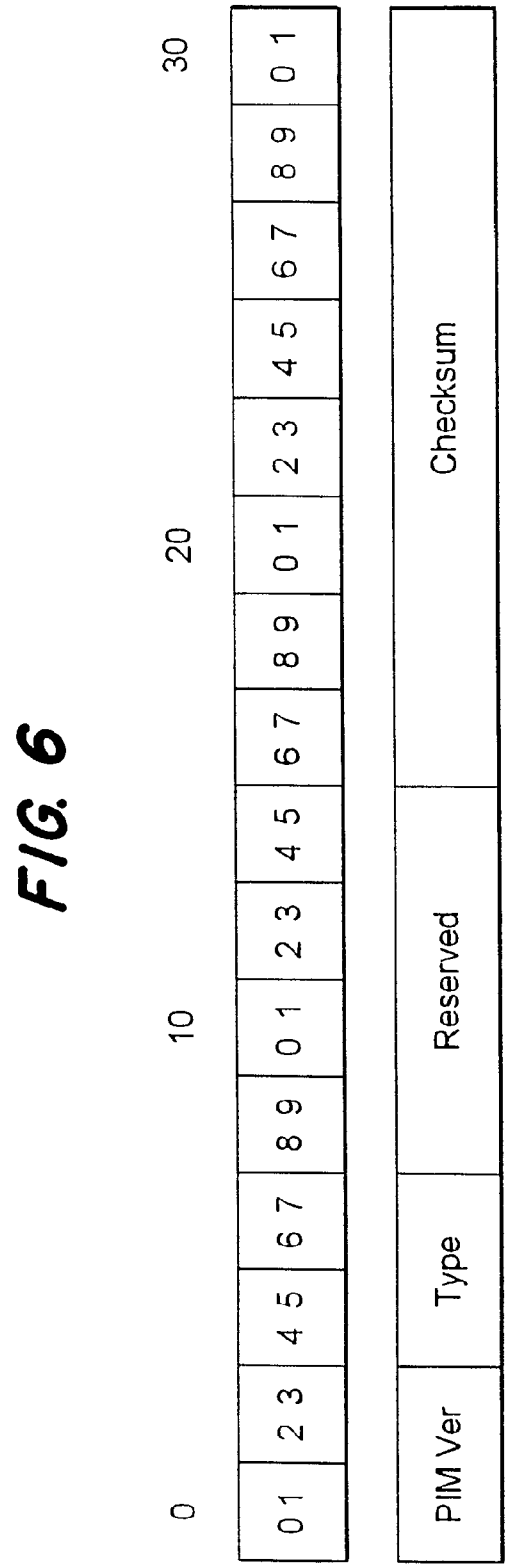
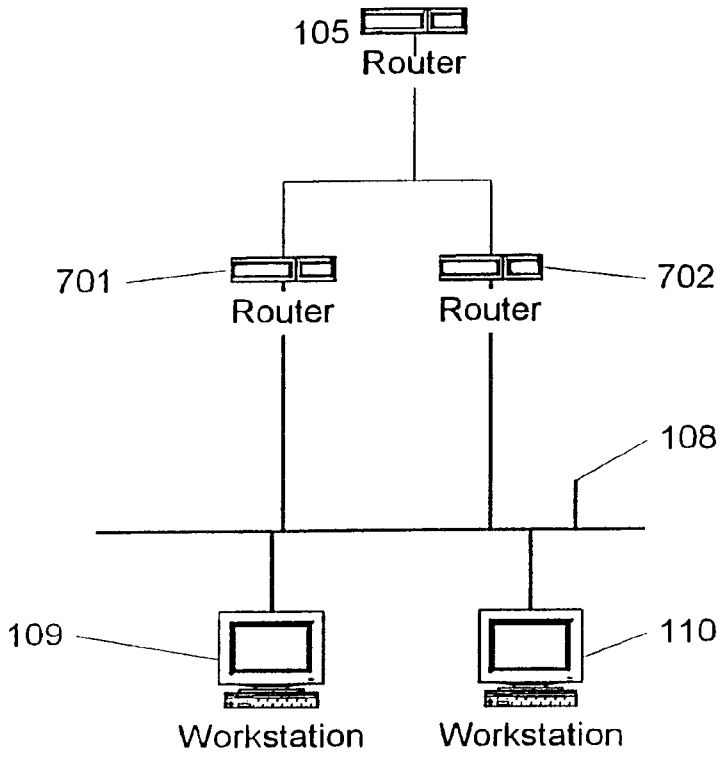

0
10
20
30
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
PIM Ver
Type
Reserved
Checksum

FIG. 7


105
Router
701
Router
Router
702
108
109
110
Workstation
Workstation

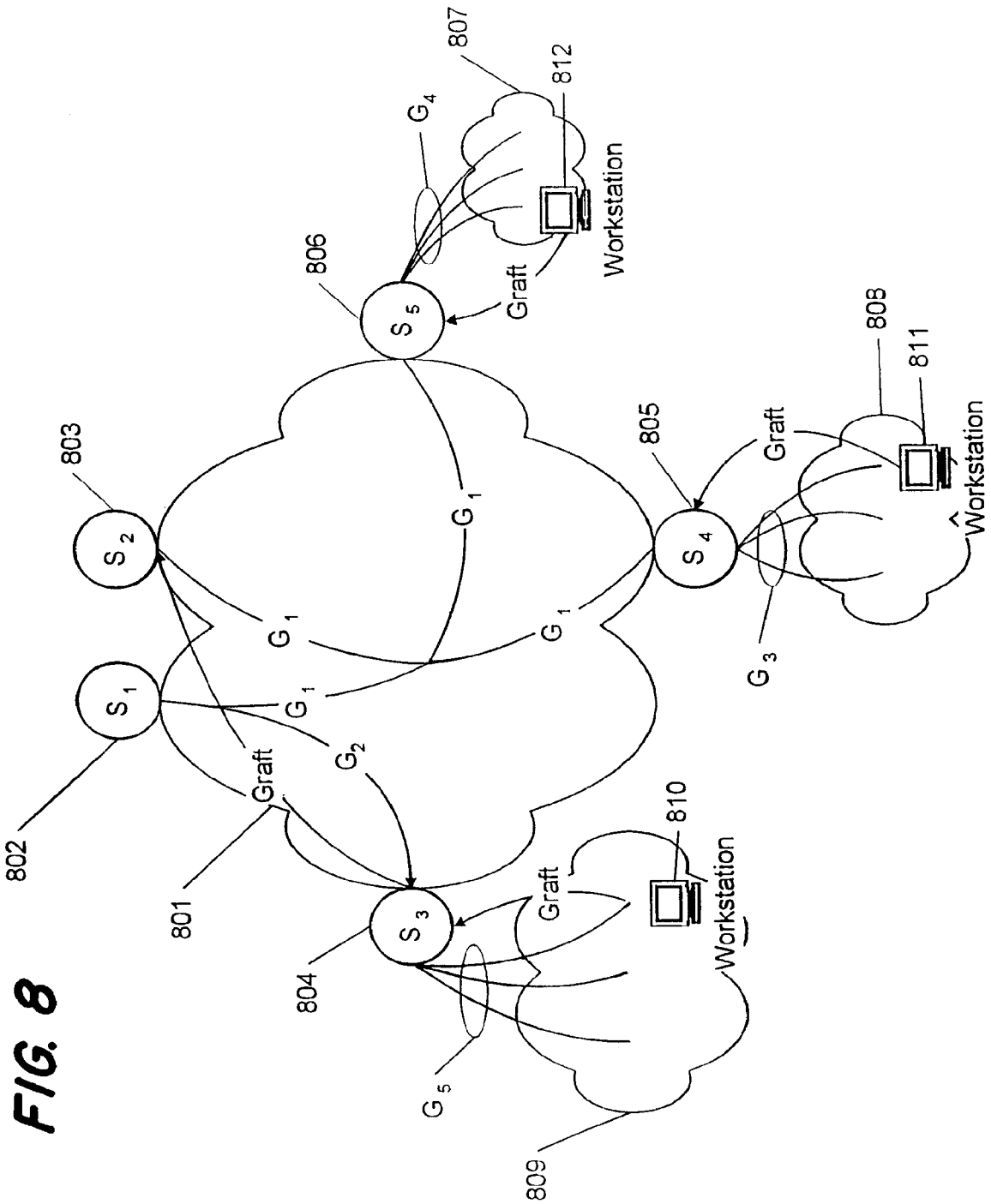

802
803
804
805
806
807
808
809
810
811
812
801
S1
S2
S3
S4
S5
G1
G2
G3
G4
G5
Graft
Workstation

FIG. 8

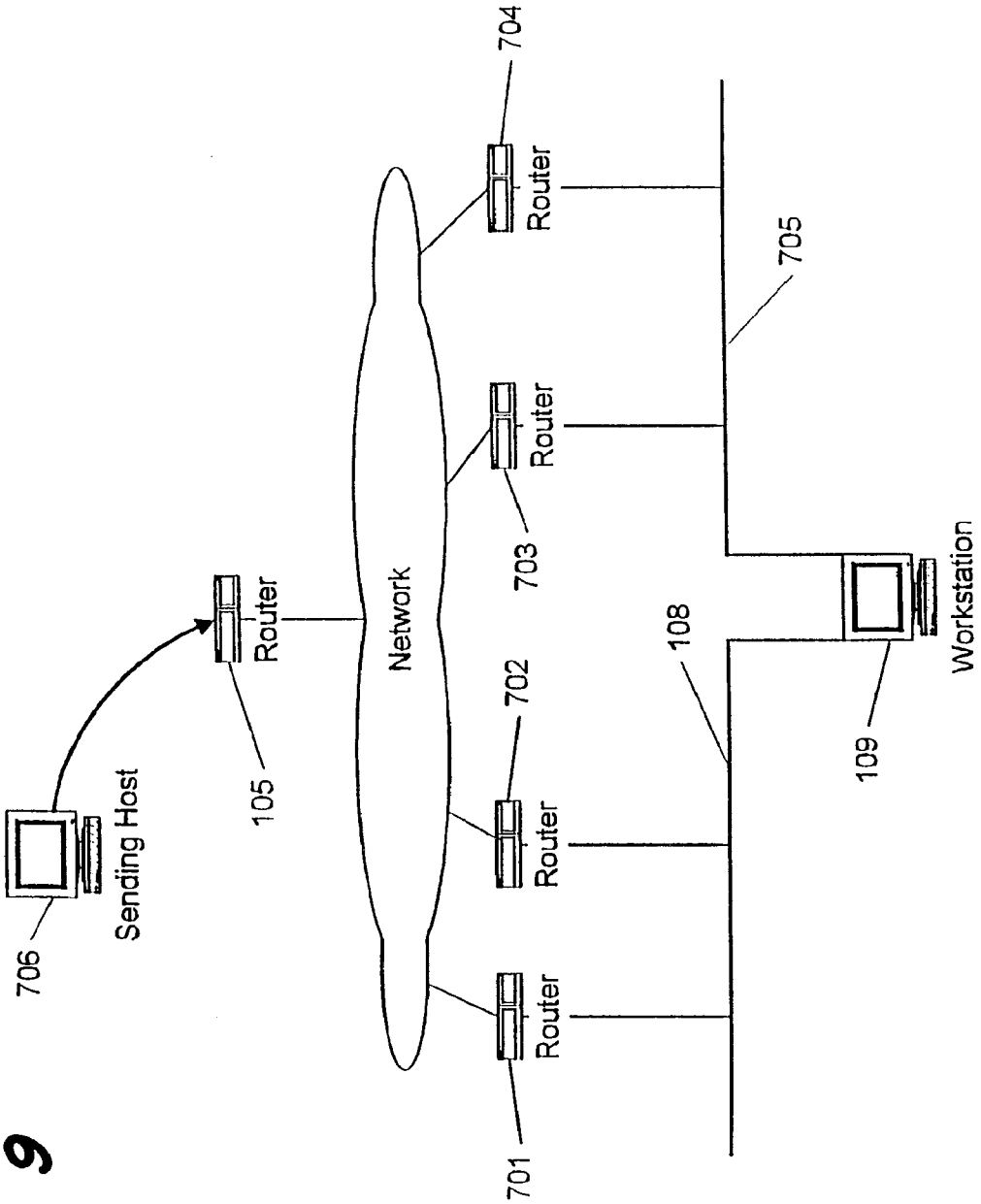

704
Router
705
Router
703
Router
Network
Workstation
108
702
Router
109
105
Router
Sending Host
706
701

FIG. 9

INTERNATIONAL SEARCH REPORT

Inte onal Application No
PCT/US 99/05731

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 H04L12/18 H04L29/14

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DEERING S ET AL: "AN ARCHITECTURE FOR WIDE-AREA MULTICAST ROUTING" COMPUTER COMMUNICATIONS REVIEW, vol. 24, no. 4, 1 October 1994 (1994-10-01), pages 126-135, XP000477046 ISSN: 0146-4833 page 129, right-hand column, line 14 - page 131, right-hand column, line 17 --- -/--	1-29

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 August 1999	30/08/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl. Fax: (+31-70) 340-3016	RAMIREZ DE AREL.., F

2

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Inter onal Application No
PCT/US 99/05731

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	BILLHARTZ T ET AL: "PERFORMANCE AND RESOURCE COST COMPARISONS FOR THE CBT AND PIM MULTICAST ROUTING PROTOCOLS" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, vol. 15, no. 3, 1 April 1997 (1997-04-01), pages 304-314, XP000683934 ISSN: 0733-8716 page 304, right-hand column, line 27 - page 306, right-hand column, line 4 page 312, right-hand column, line 38 - page 313, left-hand column, line 24 ---	1-29
A	EP 0 688 121 A (TRT TELECOM RADIO ELECTR ;PHILIPS ELECTRONICS NV (NL)) 20 December 1995 (1995-12-20) column 1, line 24 - column 2, line 8 -----	1,11,20

2

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Inter. nal Application No
PCT/US 99/05731

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0688121 A	20-12-1995	FR 2721465 A	22-12-1995
		JP 8008975 A	12-01-1996
		US 5649091 A	15-07-1997

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 6 : G06F 17/30

A1

(11) International Publication Number: WO 99/53422

(43) International Publication Date: 21 October 1999 (21.10.99)

(21) International Application Number: PCT/US99/08281

(22) International Filing Date: 15 April 1999 (15.04.99)

(30) Priority Data:
09/060,866 15 April 1998 (15.04.98) US

(71) Applicant: INKTOMI CORPORATION [US/US]; Suite 110, 1900 S. Norfolk Street, San Mateo, CA 94403–1151 (US).

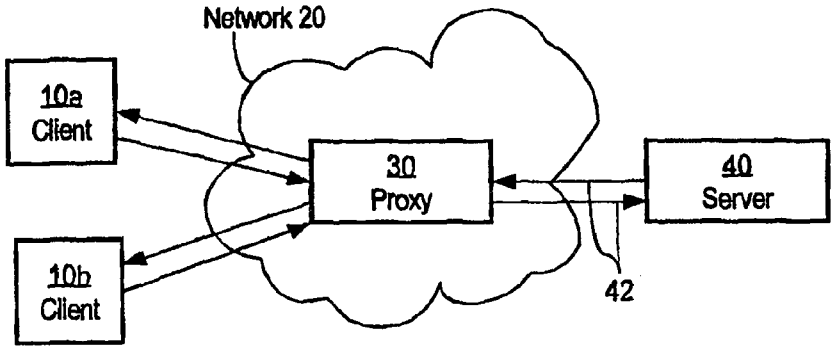
(72) Inventors: MATTIS, Peter; 1821 Hillman Avenue, Belmont, CA 94002 (US). PLEVYAK, John; 126 Baker Street, San Francisco, CA 94117 (US). HAINES, Matthew; 35 Corthell Road, Laramie, WY 82070 (US). BEGUELIN, Adam; 2821 Holland Street, San Mateo, CA 94403 (US). TOTTY, Brian; 715 Catamaran #3, Foster City, CA 94404 (US). GOURLEY, David; 2721 Midtown Court #211, Palo Alto, CA 94303 (US).

(74) Agents: CARLSON, Stephen, C. et al.; McDermott Will & Emery, 600 13th Street, N.W., Washington, DC 20005–3096 (US).

(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: HIGH PERFORMANCE OBJECT CACHE



(57) Abstract

A high–performance cache is disclosed. The cache is designed for time– and space–efficiency for a diverse range of information objects. Information objects are stored in portions of a non–volatile storage device called arenas, which are contiguous regions from which space is allocated in parallel. Objects are substantially contiguously allocated within an arena and are mapped by name keys and content–based object keys to a tag table, an open directory, and a directory table. The tag table is indexed by the name keys, and stores references to sets in the directory table. The tag table is compact and therefore can be stored in fast main memory, facilitating rapid lookups. The directory table is organized so that at least a frequently–accessed portion of it also usually resides in fast main memory, which further speeds lookups. The tag and directory tables are organized to quickly determine non–presence of objects. Large objects may be chunked into fragments, which are chained using a forward functional–iteration mechanism, to prevent the need for mutating existing on–disk data structures. Garbage collection periodically moves objects within an arena or to other arenas so that inactive objects are deleted and free space becomes contiguous. Because the objects are substantially contiguously allocated, reading and writing a typical object requires only one or two disk head actuator movements; thus, the cache can efficiently and smoothly stream data off of the storage device, providing optimal delivery of multimedia objects. The disclosure also encompasses a computer apparatus, computer program product, and computer data signal embodied in a carrier wave that are similarly configured.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

HIGH PERFORMANCE OBJECT CACHE

FIELD OF THE INVENTION

The present invention relates to information delivery, and relates more specifically to a cache for information objects that are to be delivered efficiently and at high speed over a network to a client

BACKGROUND OF THE INVENTION

Several important computer technologies rely, to a great extent, upon rapid delivery of information from a central storage location to remote devices. For example, in the client/server model of computing, one or more servers are used to store information. Client computers or processes are separated from the servers and are connected to the servers using a network. The clients request information from one of the servers by providing a network address of the information. The server locates the information based on the provided network address and transmits it over the network to the client, completing the transaction.

The World Wide Web is a popular application of the client/server computing model. Figure 1 is a simplified block diagram of the relationship between elements used in a Web system. One or more web clients 10a, 10b, each of which is a computer or a software process such as a browser program, are connected to a global information network 20 called the Internet, either directly or through an intermediary such as an Internet Service Provider, or an online information service.

A web server 40 is likewise connected to the Internet 20 by a network link 42. The web server 40 has one or more internet network addresses and textual host names, associated in an agreed-upon format that is indexed at a central Domain Name Server (DNS). The server contains multimedia information resources, such as documents and images, to be provided to clients upon demand. The server 40 may additionally or alternatively contain software for dynamically generating such resources in response to requests.

The clients 10a, 10b and server 40 communicate using one or more agreed-upon protocols that specify the format of the information that is communicated. A client 10a looks up network address of a particular server using DNS and establishes a connection to the server using a communication protocol called the Hypertext Transfer Protocol (HTTP). A Uniform Resource Locator (URL) uniquely identifies each information object stored on or dynamically generated by the server 40. A URL is a form of network address that identifies the location of information stored in a network.

A key factor that limits the performance of the World Wide Web is the speed with which the server 40 can supply information to a client via the Internet 20. Performance is limited by the speed, reliability, and congestion level of the network route through the Internet, by geographical distance delays, and by server load level. Accordingly, client transaction time can be reduced by storing replicas of popular information objects in repositories geographically dispersed from the server. Each local repository for object replicas is generally referred to as a cache. A client may be able to access replicas from a topologically proximate cache faster than possible from the original web server, while at the same time reducing Internet server traffic.

In one arrangement, as shown in Figure 1, the cache is located in a proxy server 30 that is logically interposed between the clients 10a, 10b and the server 40. The proxy server provides a "middleman" gateway service, acting as a server to the client, and a client to the server. A proxy server equipped with a cache is called a caching proxy server, or commonly, a "proxy cache".

The proxy cache 30 intercepts requests for resources that are directed from the clients 10a, 10b to the server 40. When the cache in the proxy 30 has a replica of the requested resource that meets certain freshness constraints, the proxy responds to the clients 10a, 10b and serves the resource directly. In this arrangement, the number and volume of data transfers along the link 42 are greatly reduced. As a result, network resources or objects are provided more rapidly to the clients 10a, 10b.

A key problem in such caching is the efficient storage, location, and retrieval of objects in the cache. This document concerns technology related to the storage, location, and retrieval of multimedia objects within a cache. The object storage facility within a cache is called a "cache object store" or "object store".

To effectively handle heavy traffic environments, such as the World Wide Web, a cache object store needs to be able to handle tens or hundreds of millions of different objects, while storing, deleting, and fetching the objects simultaneously. Accordingly, cache performance must not degrade significantly with object count. Performance is the driving goal of cache object stores.

Finding an object in the cache is the most common operation and therefore the cache must be extremely fast in carrying out searches. The key factor that limits cache performance is lookup time. It is desirable to have a cache that can determine whether an object is in the cache (a "hit") or not (a "miss") as fast as possible. In past approaches, caches capable of storing millions of objects have been stored in traditional file system storage structures. Traditional file systems are poorly suited for multimedia object caches because they are tuned for particular object sizes and require multiple disk head movements to examine file system metadata. Object stores can obtain higher lookup performance by dedicating DRAM memory to the task of object

lookup, but because there are tens or hundreds of millions of objects, the memory lookup tables must be very compact.

Once an object is located, it must be transferred to the client efficiently. Modern disk drives offer high performance when reading and writing sequential data, but suffer significant performance delays when incurring disk head movements to other parts of the disk. These disk head movements are called "seeks". Disk performance is typically constrained by the drive's rated seeks per second. To optimize performance of a cache, it is desirable to minimize disk seeks, by reading and writing contiguous blocks of data.

Eventually, the object store will become full, and particular objects must be expunged to make room for new content. This process is called "garbage collection". Garbage collection must be cfficient enough that it can run continually without providing a significant decrease in system performance, while removing objects that have the least impact on future cache performance.

PAST APPROACHES

In the past, four approaches have been used to structure cache object stores: using the native file system, using a memory-blocked "page" cache, using a database, and using a "cyclone" circular storage structure. Each of these prior approaches has significant disadvantages.

The native file system approach uses the file system of an operating system running on the server to create and manage a cache. File systems are designed for a particular application in mind: storing and retrieving user and system data files. File systems are designed and optimized for file management applications. They are optimized for typical data file sizes and for a relatively small number of files (both total and within one folder/directory). Traditional file systems are not optimized to minimize the number of seeks to open, read/write, and close files. Many file systems incur significant performance penalties to locate and open files when there are large numbers of files present. Typical file systems suffer fragmentation, with small disk blocks scattered around the drive surface, increasing the number of disk seeks required to access data, and wasting storage space. Also, file systems, being designed for user data file management, include facilities irrelevant to cache object stores, and indeed counter-productive to this application. Examples include: support for random access and selective modification, file permissions, support for moving files, support for renaming files, and support for appending to files over time. File systems are also invest significant energy to minimize any data loss, at the expense of performance, both at write time, and to reconstruct the file system after failure. The result is that file systems are relatively poorly for handling the millions of files that can be present in a cache of Web objects. File systems don't efficiently support the large variation in

Internet multimedia object size --- in particular they typically do not support very small objects or very large objects efficiently. File systems require a large number of disk seeks for metadata traversal and block chaining, poorly support garbage collection, and take time to ensure data integrity and to repair file systems on restart.

The page cache extends file systems with a set of fixed sized memory buffers. Data is staged in and out of these buffers before transmission across the network. This approach wastes significant memory for large objects being sent across slow connections.

The database system approach uses a database system as a cache. Generally, databases are structured to achieve goals that make them inappropriate for use as an object cache. For example, they are structured to optimize transaction processing. To preserve the integrity of each transaction, they use extensive locking. As a result, as a design goal they favor data integrity over performance factors such as speed. In contrast, it is acceptable for an object cache to lose data occasionally, provided that the cache does not corrupt objects, because the data always can be retrieved from the server that is original source of the data. Databases are often optimized for fast write performance, since write speed limits transaction processing speed. However, in an object cache, read speed is equally important. Further, databases are not naturally good at storing a vast variety of object sizes while supporting streaming, pipelined I/O in a virtual memory efficient manner. Databases commonly optimized for fixed record size sizes. Where databases support variable record sizes, they contain support for maintaining object relationships that are redundant, and typically employ slow, virtual memory paging techniques to support streaming, pipelined I/O.

In a cyclonic file system, data is allocated around a circular storage structure. When space becomes full, the oldest data is simply removed. This approach allows for fast allocation of data, but makes it difficult to support large objects without first staging them in memory, suffers problems with fragmentation of data, and typically entails naïve garbage collection that throws out the oldest object, regardless of its popularity. For a modest, active cache with a diverse working set, such first-in-first-out garbage collection can throw objects out before they get to be reused.

The fundamental problem with the above approaches for the design of cache object stores is that the solution isn't optimized for the constraints of the problem. These approaches all represent reapplication of existing technologies to a new application. None of the applications above are ideally suited for the unique constraints of multimedia, streaming, object caches. Not only do the above solutions inherently encumber object caches with inefficiencies due to their imperfect reapplication, but they also are unable to effectively support the more unique requirements of multimedia object caches. These unique requirements include the ability to

disambiguate and share redundant content that is identical, but has different names, and the opposite ability to store multiple variants of content with the same name, targeted for particular clients, languages, data types, etc.

Based on the foregoing, there is a clear need to provide an object cache that overcomes the disadvantages of these prior approaches, and is more ideally suited for the unique requirements of multimedia object caches. In particular:

1. there is a need for an object store that can store hundreds of millions of objects of disparate sizes, and a terabyte of content size in a memory efficient manner;
2. there is a need for an object store that can determine if a document is a "hit" or a "miss" quickly, without time-consuming file directory lookups;
3. there is a need for a cache that minimizes the number of disk seeks to read and write objects;
4. there is a need for an object store that permits efficient streaming of data to and from the cache;
5. there is a need for an object store that supports multiple different versions of targeted alternates for the same name;
6. there is a need for an object store that efficiently stores large numbers of objects without content duplication;
7. there is a need for an object store that can be rapidly and efficiently garbage collected in real-time, insightfully selecting the documents to be replaced to improve user response speed, and traffic reduction;
8. there is a need for an object store that that can restart to full operational capacity within seconds after software or hardware failure without data corruption and with minimal data loss.

This document concerns technology directed to accomplishing the foregoing goals. In particular, this document describes methods and structures related to the time-efficient and space-efficient storage, retrieval, and maintenance of objects in a large object store. The technology described herein provides for a cache object store for a high-performance, high-load application having the following general characteristics:

1. High performance, measured in low latency and high throughput for object store operations, and large numbers of concurrent operations;
2. Large cache support, supporting terabyte caches and billions of objects, to handle the Internet's exponential content growth rate;
3. Memory storage space efficiency, so expensive semiconductor memory is used sparingly and effectively;

4. Disk storage space efficiency, so large numbers of Internet object replicas can be stored within the finite disk capacity of the object store;
5. Alias free, so that multiple objects or object variants, with different names, but with the same content identical object content, will have the object content cached only once, shared among the different names;
6. Support for multimedia heterogeneity, efficiently supporting diverse multimedia objects of a multitude of types with size ranging over six orders of magnitude from a few hundred bytes to hundreds of megabytes;
7. Fast, usage-aware garbage collection, so less useful objects can be efficiently removed from the object store to make room for new objects;
8. Data consistency, so programmatic errors and hardware failures do not lead to corrupted data;
9. Fast restartability, so an object cache can begin servicing requests within seconds of restart, without requiring a time-consuming database or file system check operation;
10. Streaming, so large objects can be efficiently pipelined from the object store to slow clients, without staging the entire object into memory;
11. Support for content negotiation, so proxy caches can efficiently and flexibly store variants of objects for the same URL, targeted on client browser, language, or other attribute of the client request; and
12. General-purpose applicability, so that the object store interface is sufficiently flexible to meet the needs of future media types and protocols.

SUMMARY OF THE INVENTION

The foregoing needs and other needs are addressed by the present invention, which provides, in one aspect, in a cache for information objects that are identified by key values based on names of the information objects, comprising a tag table that indexes the information objects using set subkey values based on the key values, a directory table having a plurality of blocks indexed to sets in the tag table by second subkey values based on the key values, and data storage areas referenced by the blocks in the directory table, a method of delivering a requested information object to a client from the cache at a server, comprising the steps of receiving a name that identifies a requested information object; computing a fixed size key value comprising a plurality of subkeys, based on the name; looking up the requested information object in a directory table, using the subkeys as lookup keys; and retrieving a copy of the requested information object from the data storage areas using a reference contained in a matching block in the directory table.

A feature of this aspect involves the steps of selecting a version of the requested information object from a list in said cache of a plurality of versions of the requested information objects; identifying a storage location of the requested information object in said cache based on an object key stored in the list in association with the first version; retrieving the requested information object from the storage location; and delivering the requested information object to the client.

Another feature involves storing the information objects contiguously in a mass storage device. Yet another feature is storing each of the information objects in a contiguous pool of the mass storage device. Still another feature is storing each of the information objects in one of a plurality of arenas in the pool. Another feature is storing each of the information objects in one or more fragments, allocated from arenas.

According to another feature, the fragments comprise an information object are linked from the previous fragment key. Another feature involves storing the list contiguously with each of the plurality of versions of the requested information object; in each of the blocks, storing a size value of the requested information object in association with such block, wherein the size value indicates a storage size of the list and the plurality of versions of the information object; and wherein step (D) comprises the step of reading the list and the plurality of versions concurrently. Yet another feature is consolidating streaming data transfers of different speeds into a write aggregation buffer.

According to still another feature, the step of storing the information objects comprises the step of writing the information objects in contiguous available storage space of the mass

storage device, while concurrently performing steps (A) through (D) with respect to another information object.

The invention also encompasses an apparatus, computer system, computer program product, and a computer data signal embodied in a carrier wave configured according to the foregoing aspects, and other aspects.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is illustrated by way of example, and not by way of limitation, in the figures of the accompanying drawings and in which like reference numerals refer to similar elements and in which:

Figure 1 is a block diagram of a client/server relationship;

Figure 2 is a block diagram of a traffic server;

Figure 3A is a block diagram of transformation of an object into a key;

Figure 3B is a block diagram of transformation of an object name into a key;

Figure 4A is a block diagram of a cache;

Figure 4B is a block diagram of a storage mechanism for Vectors of Alternates;

Figure 4C is a block diagram of multi-segment directory table;

Figure 5 is a block diagram of pointers relating to data fragments;

Figure 6 is a block diagram of a storage device and its contents;

Figure 7 is a block diagram showing the structure of a pool;

Figure 8A is a flow diagram of a process of garbage collection;

Figure 8B is a flow diagram of a process of writing information in a storage device;

Figure 8C is a flow diagram of a process of synchronization;

Figure 8D is a flow diagram of a "checkout_read" process;

Figure 8E is a flow diagram of a "checkout_write" process;

Figure 8F is a flow diagram of a "checkout_create" process;

Figure 9A is a flow diagram of a cache lookup process;

Figure 9B is a flow diagram of a "checkin" process;

Figure 9C is a flow diagram of a cache lookup process;

Figure 9D is a flow diagram of a cache remove process;

Figure 9E is a flow diagram of a cache read process;

Figure 9F is a flow diagram of a cache write process;

Figure 9G is a flow diagram of a cache update process;

Figure 10A is a flow diagram of a process of allocating and writing objects in a storage device;

Figure 10B is a flow diagram of a process of scaled counter updating;

Figure 11 is a block diagram of a computer system that can be used to implement the present invention;

Figure 12 is a flow diagram of a process of object re-validation.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

A method and apparatus for caching information objects is described. In the following description, for the purposes of explanation, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known structures and devices are shown in block diagram form in order to avoid unnecessarily obscuring the present invention.

TRAFFIC SERVER

Figure 2 is a block diagram of the general structure of certain elements of a proxy 30. In one embodiment, the proxy 30 is called a traffic server and comprises one or more computer programs or processes that operate on a computer workstation of the type described further below. A client 10a directs a request 50 for an object to the proxy 30 via the Internet 20. In this context, the term "object" means a network resource or any discrete element of information that is delivered from a server. Examples of objects include Web pages or documents, graphic images, files, text documents, and objects created by Web application programs during execution of the programs, or other elements stored on a server that is accessible through the Internet 20. Alternatively, the client 10a is connected to the proxy 30 through a network other than the Internet.

The incoming request 50 arrives at an input/output (I/O) core 60 of the proxy 30. The I/O core 60 functions to adjust the rate of data received or delivered by the proxy to match the data transmission speed of the link between the client 10a and the Internet 20. In a preferred embodiment, the I/O core 60 is implemented in the form of a circularly arranged set of buckets that are disposed between input buffers and output buffers that are coupled to the proxy 30 and the Internet 20. Connections among the proxy 30 and one or more clients 10a are stored in the buckets. Each bucket in the set is successively examined, and each connection in the bucket is polled. During polling, the amount of information that has accumulated in a buffer associated with the connection since the last poll is determined. Based on the amount, a period value associated with the connection is adjusted. The connection is then stored in a different bucket that is generally identified by the sum of the current bucket number and the period value. Polling continues with the next connection and the next bucket. In this way, the elapsed time between successive polls of a connection automatically adjusts to the actual operating bandwidth or data communication speed of the connection.

The I/O core 60 passes the request 50 to a protocol engine 70 that is coupled to the I/O core 60 and to a cache 80. The protocol engine 70 functions to parse the request 50 and

determine what type of substantive action is embodied in the request 50. Based on information in the request 50, the protocol engine 70 provides a command to the cache 80 to carry out a particular operation. In an embodiment, the cache 80 is implemented in one or more computer programs that are accessible to the protocol engine 70 using an application programming interface (API). In this embodiment, the protocol engine decodes the request 50 and performs a function call to the API of the cache 80. The function call includes, as parameter values, information derived from the request 50.

The cache 80 is coupled to send and receive information to and from the protocol engine 70 and to interact with one or more non-volatile mass storage devices 90a-90n. In an embodiment, the storage devices 90a-90n are high-capacity, fast disk drives. The cache 80 also interacts with data tables 82 that are described in more detail herein.

OBJECT CACHE INDEXING

CONTENT INDEXING

In the preferred embodiment, the cache 80 stores objects on the storage devices 90a-90n. Popular objects are also replicated into a cache. In the preferred embodiment, the cache has finite size, and is stored in main memory or RAM of the proxy 30.

Objects on disk are indexed by fixed sized locators, called keys. Keys are used to index into directories that point to the location of objects on disk, and to metadata about the objects. There are two types of keys, called "name keys" and "object keys". Name keys are used to index metadata about a named object, and object keys are used to index true object content. Name keys are used to convert URLs and other information resource names into a metadata structure that contains object keys for the object data. As will be discussed subsequently, this two-level indexing structure facilitates the ability to associate multiple alternate objects with a single name, while at the same time maintaining a single copy of any object content on disk, shared between multiple different names or alternates.

Unlike other cache systems that use the name or URL of an object as the key by which the object is referenced, embodiments of the invention use a "fingerprint" of the content that makes up the object itself, to locate the object. Keys generated from the content of the indexed object are referred to herein as object keys. Specifically, the object key 56 is a unique fingerprint or compressed representation of the contents of the object 52. Preferably, a copy of the object 52 is provided as input to a hash function 54, and its output is the object key 56. For example, a file or other representation of the object 52 is provided as input to the hash function, which reads each byte of the file and generates a portion of the object key 56, until the entire file has been read. In this way, an object key 56 is generated based upon the entire contents of the object 52

rather than its name. Since the keys are content-based, and serve as indexes into tables of the cache 80, the cache is referred to as a content-indexed cache. Given a content fingerprint key, the content can easily be found.

In this embodiment, content indexing enables the cache 80 to detect duplicate objects that have different names but the same content. Such duplicates will be detected because objects having identical content will hash to the same key value even if the objects have different names.

For example, assume that the server 40 is storing, in one subdirectory, a software program comprising an executable file that is 10 megabytes in size, named "IE4.exe". Assume further that the server 40 is storing, in a different subdirectory, a copy of the same file, named "Internet Explorer.exe". The server 40 is an anonymous FTP server that can deliver copies of the files over an HTTP connection using the FTP protocol. In past approaches, when one or more clients request the two files, the cache stores a copy of each of the files in cache storage, and indexes each of the files under its name in the cache. As a result, the cache must use 20 megabytes of storage for two objects that are identical except for the name.

In embodiments of the invention, as discussed in more detail herein, for each of the objects, the cache creates a name key and an object key. The name keys are created by applying a hash function to the name of the object. The object keys are created by applying a hash function to the content of the object. As a result, for the two exemplary objects described above, two different name keys are created, but the object key is the same. When the first object is stored in the cache, its name key and object key are stored in the cache. When the second object is stored in the cache thereafter, its name key is stored in the cache. However, the cache detects the prior identical object key entry, and does not store a duplicate object key entry; instead, the cache stores a reference to the same object key entry in association with the name key, and deletes the new, redundant object. As a result, only 10 megabytes of object storage is required. Thus, the cache detects duplicate objects that have different names, and stores only one permanent copy of each such object.

Figure 3A is a block diagram of mechanisms used to generate an object key 56 for an object 52. When client 10a requests an object 52, and the object is not found in the cache 80 using the processes described herein, the cache retrieves the object from a server and generates a object key 56 for storing the object in the cache.

Directories are the data structures that map keys to locations on disk. It is advisable to keep all or most of the contents of the directories in memory to provide for fast lookups. This requires directory entries to be small, permitting a large number of entries in a feasible amount of memory. Further, because 50% of the accesses are expected not to be stored in cache, we want to determine cache misses quickly, without expending precious disk seeks. Such fast miss optimizations dedicate scarce disk head movements to real data transfers, not unsuccessful speculative lookups. Finally, to make lookups fast via hashing search techniques, directory entries are fixed size.

Keys are carefully structured to be fixed size and small, for the reasons described earlier. Furthermore, keys are partitioned into subkeys for the purposes of storage efficiency and fast lookups. Misses can be identified quickly by detecting differences in just a small portion of keys. For this reason, instead of searching a full directory table containing complete keys, misses are filtered quickly using a table of small subkeys called a "tag table". Furthermore, statistical properties of large bit vectors can be exploited to create space-efficient keys that support large numbers of cache objects with small space requirements.

According to one embodiment, the object key 56 comprises a set subkey 58 and a tag subkey 59. The set subkey 58 and tag subkey 59 comprise a subset of the bits that make up the complete object key 56. For example, when the complete object key 56 is 128 bits in length, the subkeys 58, 59 can be 16 bits, 27 bits, or any other portion of the complete key. The subkeys 58, 59 are used in certain operations, which are described below, in which the subkeys yield results that are nearly as accurate as when the complete key is used. In this context, "accurate" means that use of the subkeys causes a hit in the cache to the correct object as often as when the complete key is used.

This accuracy property is known as "smoothness" and is a characteristic of a certain preferred subset of hash functions. An example of a hash function suitable for use in an embodiment is the MD5 hash function, which is described in detail in B. Schneier, "Applied Cryptography" (New York: John Wiley & Sons, Inc., 2d ed. 1996), at pp. 429-431 and pp. 436-441. The MD5 hash function generates a 128-bit key from an input data stream having an arbitrary length. Generally the MD5 hash function and other one-way hash functions are used in the cryptography field to generate secure keys for messages or documents that are to be transmitted over secure channels. General hashing table construction and search techniques are

described in detail in D. Knuth, "The Art of Computer Programming: Vol. 3, Sorting and Searching," at 506-549 (Reading, MA: Addison-Wesley, 1973).

NAME INDEXING

Unfortunately, requests for objects typically do not identify requested objects using the object keys for the objects. Rather, requests typically identify requested objects by name. The format of the name may vary from implementation to implementation based on the environment in which the cache is used. For example, the object name may be a file system name, a network address, or a URL.

According to one aspect of the invention, the object key for a requested object is indexed under a "name key" that is generated based on the object name. Thus, retrieval of an object in response to a request is a two phase process, where a name key is used to locate the object key, and the object key is used to locate the object itself.

Figure 3B is a block diagram of mechanisms used to generate a name key 62 based on an object name 53. According to one embodiment, the same hash function 54 that is used to generate object keys is used to generate name keys. Thus, the name keys will have the same length and smoothness characteristics of the object keys.

Similar to object key 56, the name key 62 comprises set and tag subkeys 64, 66. The subkeys 64, 66 comprise a subset of the bits that make up the complete name key 62. For example, when the complete name key 62 is 128 bits in length, the first and second subkeys 64, 66 can be 16 bits, 27 bits, or any other portion of the complete key.

SEARCHING BY OBJECT OR NAME KEY

Preferably, the cache 80 comprises certain data structures that are stored in the memory of a computer system or in its non-volatile storage devices, such as disks. Figure 4 is a block diagram of the general structure of the cache 80. The cache 80 generally comprises a Tag Table 102, a Directory Table 110, an Open Directory table 130, and a set of pools 200a through 200n, coupled together using logical references as described further below.

The Tag Table 102 and the Directory Table 110 are organized as set associative hash tables. The Tag Table 102, the Directory Table 110, and the Open Directory table 130 correspond to the tables 82 shown in Figure 2. For the purposes of explanation, it shall be assumed that an index search is being performed based on object key 56. However, the Tag Table 102 and Directory Table 110 operate in the same fashion when traversed based on a name key 62.

The Tag Table 102 is a set-associative array of sets 104a, 104b, through 104n. The tag table is designed to be small enough to fit in main memory. Its purpose is to quickly detect misses, whereby using only a small subset of the bits in the key a determination can be made that the key is not stored in the cache. The designation 104n is used to indicate that no particular number of sets is required in the Tag Table 102. As shown in the case of set 104n, each of the sets 104a-104n comprises a plurality of blocks 106.

In the preferred embodiment, the object key 56 is 128 bits in length. The set subkey 58 is used to identify and select one of the sets 104a-104n. Preferably, the set subkey 58 is approximately 18 bits in length. The tag subkey 59 is used to reference one of the entries 106 within a selected set. Preferably, the tag subkey 59 is approximately 16 bits in length, but may be as small as zero bits in cases in which there are many sets. In such cases, the tag table would be a bit vector.

The mechanism used to identify or refer to an element may vary from implementation to implementation, and may include associative references, pointers, or a combination thereof. In this context, the term "reference" indicates that one element identifies or refers to another element. A remainder subkey 56' consists of the remaining bits of the key 56. The set subkey, tag subkey, and remainder subkey are sometimes abbreviated *s*, *t*, and *r*, respectively.

The preferred structure of the Tag Table 102, in which each entry contains a relatively small amount of information enables the Tag Table to be stored in fast, volatile main memory such as RAM. Thus, the structure of the Tag Table 102 facilitates rapid operation of the cache. The blocks in the Directory Table 110, on the other hand, include much more information as described below, and consequently, portions of the Directory Table may reside on magnetic disk media as opposed to fast DRAM memory at any given time.

The Directory Table 110 comprises a plurality of sets 110a-110n. Each of the sets 110a-110n has a fixed size, and each comprises a plurality of blocks 112a-112n. In the preferred embodiment, there is a predetermined, constant number of sets and a predetermined, constant number of blocks in each set. As shown in the case of block 112n, each of the blocks 112a-112n stores a third, remainder subkey value 116, a disk location value 118, and a size value 120. In the preferred embodiment, the remainder subkey value 116 is a 27-bit portion of the 128-bit complete object key 56, and the comprises bits of the complete object key 56 that are disjoint from the bits that comprise the set or tag subkeys 58, 59.

In a search, the subkey values stored in the entry 106 of the Tag Table 102 matches or references one of the sets 110a-110n, as indicated by the arrow in Figure 4 that connects the entry 106 to the set 110d. As an example, consider the 12-bit key and four-bit first and second subkeys described above. Assume that the set subkey value 1111 matches set 104n of the Tag

Table 102, and the tag subkey value 0000 matches entry 106 of set 104n. The match of the tag subkey value 0000 indicates that there is a corresponding entry in set 110d of the Directory Table 110 associated with the key prefix 11110000. When one of the sets 110a-110n is selected in this manner, the blocks within the selected set are searched linearly to find a block, such as block 112a, that contains the remainder subkey value 116 that matches a corresponding portion of the object key 56. If a match is found, then there is almost always a hit in the cache. There is a small possibility of a miss if the first, second and third subkeys don't comprise the entire key. If there is a hit, the referenced object is then located based on information contained in the block, retrieved from one of the cache storage devices 90a-90n, and provided to the client 10a, as described further below.

Unlike the Tag Table, whose job is to quickly determine rule out misses with the minimal use of RAM memory, each block within Directory Table 110 includes a full pointer to a disk location. The item referenced by the disk location value 118 varies depending on the source from which the key was produced. If the key was produced based on the content of an object, as described above, then the disk location value 118 indicates the location of a stored object 124 (or a first fragment thereof), as shown in Figure 4 in the case of block 112b. If the key is a name key, then as shown for block 112n, the disk location value 118 indicates the location of one or more Vectors of Alternates 122, each of which stores one or more object keys for the object whose name was used to generate the name key. A single Tag Table 102 and a single Directory Table 110 are shown in Figure 4 merely by way of example. However, additional tables that provide additional levels of storage and indexing may be employed in alternate embodiments.

In the preferred arrangement, when a search of the cache is conducted, a hit or miss will occur in the Tag Table 102 very quickly. If there is a hit in the Tag Table 102, then there is a very high probability that a corresponding entry will exist in the Directory Table 110. The high probability results from the fact that a hit in the Tag Table 102 means that the cache holds an object whose full key shares X identical bits to the received key, where X is the number of bits of the concatenation of the set and tag subkeys 58 and 59. Because misses can be identified quickly, the cache 80 operates rapidly and efficiently, because hits and misses are detected quickly using the Tag Table 102 in memory without requiring the entire Directory Table 110 to reside in main memory.

When the cache is searched based on object key 56, the set subkey 58 is used to index one of the sets 104a-104n in Tag Table 102. Once the set associated with subkey 58 is identified, a linear search is performed through the elements in the set to identify an entry whose tag matches the tag subkey 59.

In a search for an object 52 requested from the cache 80 by a client 10a, when one of the sets 104a-104n is selected using the set subkey 58, a linear search of all the elements 106 in that set is carried out. The search seeks a match of the tag subkey 59 to one the entries. If a match is found, then there is a hit in the Tag Table 102 for the requested object, and the cache 80 proceeds to seek a hit in the Directory Table 110.

For purposes of example, assume that the object key is a 12-bit key having a value of 111100001010, the set subkey comprises the first four bits of the object key having a value of 1111, and the tag subkey comprises the next four bits of the object key having a value of 0000. In production use the number of remainder bits would be significantly larger than the set and tag bits to affect memory savings. The cache identifies set 15 (1111) as the set to examine in the Tag Table 102. The cache searches for an entry within that set that contains a tag 0000. If there is no such entry, then a miss occurs in the Tag Table 102. If there is such an entry, then the cache proceeds to check the remaining bits in Directory Table 110 for a match.

MULTI-LEVEL DIRECTORY TABLE

In one embodiment, the Directory Table 110 contains multiple sets each composed of a fixed number of elements. Each element contains the remainder tag and a disk pointer. Large caches will contain large numbers of objects, which will require large numbers of elements in the directory table. This can create tables too large to be cost-effectively stored in main memory.

For example, if a cache was configured with 128 million directory table elements, and each element was represented by a modest 8 bytes of storage, 1 GByte of memory would be requires to store the directory table, which is more memory than is common on contemporary workstation computers. Because few of these objects will be actively accessed at any time, there is a desire to migrate the underutilized entries onto disk while leaving higher utilized entries in main memory.

Figure 4C is a diagram of a multi-level directory mechanism. The directory table 110 is partitioned into segments 111a, 111b, 111c. In the preferred embodiments, there are two or three segments 111a-111c, although a larger number of segments may be used. The first segment 111a is the smallest, and fits in main memory such as the main memory 1106 of the computer system shown in Figure 11 and discussed in detail below. The second and third segments 111b, 111c are progressively larger. The second and third segments 111b, 111c are coupled through a paging mechanism to a mass storage device 1110 such as a disk. The second and third segments 111b, 111c dynamically page data in from the disk if requested data is not present in the main memory 1106.

As directory elements are accessed more often, the directory elements are moved to successively higher segment among the segments 111a-111c of the multi-level directory. Thus, frequently accessed directory elements are more likely to be stored in main memory 1106. The most popular elements appear in the highest and smallest segment 111a of the directory, and will all be present in main memory 1106. Popularity of entries is tracked using a small counter that is several bits in length. This counter is updated as described in the section SCALED COUNTER UPDATING. This multi-level directory approximates the performance of in-memory hash tables, while providing cost-effective aggregate storage capacity for terabyte-sized caches, by placing inactive elements on disk.

DIRECTORY PAGING

As discussed, in a preferred embodiment, the Directory Table 110 is implemented as a multi-level hash table. Portions of the Directory Table may reside out of main memory, on disk. Data for the Directory Table is paged in and out of disk on demand. A preferred embodiment of this mechanism uses direct disk I/O to carefully control the timing of paging to and from disk and the amount of information that is paged.

Another embodiment of this approach exploits a feature of UNIX-type operating systems to map files directly into virtual memory segments. In this approach, the cache maps the Directory Table into virtual memory using the UNIX mmap() facility. For example, a mmap request is provided to the operating system, with a pointer to a file or disk location as a parameter. The mmap request operates as a request to map the referenced file or disk location to a memory location. Thereafter, the operating system automatically loads portions of the referenced file or disk location from disk into memory as necessary.

Further, when the memory location is updated or accessed, the memory version of the object is written back to disk as necessary. In this way, native operating system mechanisms are used to manage backup storage of the tables in non-volatile devices. However, at any given time it is typical that only a portion of the Directory Table 110 is located in main memory.

In a typical embodiment, the Directory Table and Open Directory are stored using a "striping" technique. Each set of the tables is stored on a different physical disk drive. For example, set 110a of Directory Table 110 is stored on storage device 90a, set 110b is stored on storage device 110b, etc. In this arrangement, the number of seek operations needed for a disk drive head to arrive at a set is reduced, thereby improving speed and efficiency of the cache.

It should be noted when paging data between disk and memory certain safeguards are taken to ensure that the information stored in memory is consistent with the corresponding information stored in a non-volatile storage device. The techniques used to provide efficient

consistency in object caches are summarized in the context of garbage collection, in the section named SYNCHRONIZATION AND CONSISTENCY ENFORCEMENT.

VECTOR OF ALTERNATES

As mentioned above, it is possible for a single URL to map to an object that has numerous versions. These versions are called "alternates". In systems that do not use an object cache, versions are selected as follows. The client 10a establishes an HTTP connection to the server 40 through the Internet 20. The client provides information about itself in an HTTP message that requests an object from the server. For example, an HTTP request for an object contains header information that identifies the Web browser used by the client, the version of the browser, the language preferred by the client, and the type of media content preferred by the client. When the server 40 receives the HTTP request, it extracts the header information, and selects a variant of the object 52 based upon the values of the header information. The selected alternate is returned to the client 10a in a response message. This type of variant selection is promoted by the emerging HTTP/1.1 hypertext transfer protocol.

It is important for a cache object store to efficiently maintain copies of alternates for a URL. If a single object is always served from cache in response to any URL requests, a browser may receive content that is different than that obtained directly from a server. For this reason, each name key in the directory table 110 maps to one of the vectors of alternates 122a-122n, which enable the cache 80 to select one version of an object from among a plurality of related versions. For example, the object 52 may be a Web page and server 40 can store versions of the object in the English, French, and Japanese languages.

Each Vector of Alternates 122a-122n is a structure that stores a plurality of alternate records 123a-123n. Each of the alternate records 123a-123n is a structure that stores information that describes an alternative version of the requested object 52. For example the information describes a particular browser version, a human language in which the object has been prepared, etc. The alternate records also each store a full object key that identifies an object that contains the alternative version. In the preferred embodiment, each of the alternate records 123a-123n stores request information, response information, and an object key 56.

Because a single popular object name may map to many alternates, in one embodiment a cache composes explicit or implicit request context with the object name to reduce the number of elements in the vector. For example, the User-Agent header of a Web client request (which indicates the particular browser application) may be concatenated with a web URL to form the name key. By including contextual information directly in the key, the number of alternates in each vector is reduced, at the cost of more entries in the directory table. In practice, the

particular headers and implicit context concatenated with the information object name is configurable.

These Vectors of Alternates 122a-122n support the correct processing of HTTP/1.1 negotiated content. Request and response information contained in the headers of HTTP/1.1 messages is used to determine which of the alternate records 123a-123n can be used to satisfy a particular request. When cache 80 receives requests for objects, the requests typically contain header information in addition to the name (or URL) of the desired object. As explained above, the name is used to locate the appropriate Vector of Alternates. Once the appropriate Vector of Alternates is found, the header information is used to select the appropriate alternate record for the request.

Specifically, in the cache 80, the header information is received and analyzed. The cache 80 seeks to match values found in the header information with request information of one of the alternate records 123a-123n. For example, when the cache 80 is used in the context of the World Wide Web, requests for objects are provided to a server containing the cache in the form of HTTP requests.

The cache 80 examines information in an HTTP request to determine which of the alternate records 123a-123n to use. For example, the HTTP request might contain request information indicating that the requesting client 10a is running the Netscape Navigator browser program, version 3.0, and prefers German text. Using this information, the cache 80 searches the alternate records 123a through 123n for response information that matches the browser version and the client's locale from the request information. If a match is found, then the cache retrieves the object key from the matching alternate and uses the object key to retrieve the corresponding object from the cache.

The cache optimizes the object chosen by matching the criteria specified in the client request. The client request may specify minimal acceptance criteria (e.g. the document must be a JPEG image, or the document must be Latin). The client request may also specify comparative weighting criteria for matches (e.g. will accept a GIF image with weight 0.5, but prefer a JPEG image at weight 0.75). The numeric weightings are accumulated across all constraint axes to create a final weighting that is optimized.

The object key is used to retrieve the object in the manner described above. Specifically, a subkey portion of the object key is used to initiate another search of the Tag Table 102 and the Directory Table 110, seeking a hit for the subkey value. If there is a hit in both the Tag and Directory Tables, then the block in the Directory Table arrived at using the subkey values will always reference a stored object (e.g. stored object 124). Thus, using the Vector of Alternates

122, the cache 80 can handle requests for objects having multiple versions and deliver the correct version to the requesting client 10a.

In Figure 4, only one exemplary Vector of Alternates 122 and one exemplary stored object 124 are shown. However, in practice the cache 80 includes any number of vectors and disk blocks, depending on the number of objects that are indexed and the number of alternative versions associated with the objects.

READ AHEAD

Figure 4B is a diagram showing a storage arrangement for exemplary Vectors of Alternates 122a-122n. The system attempts to aggregate data object contiguously after the metadata. Because seeks are time-consuming but sequential reads are fast, performance is improved by consolidating data with metadata, and pre-fetching data after the metadata.

In one of the storage devices 90a-90n, each of the Vectors of Alternates 122a-122n is stored in a location that is contiguous to the stored objects 124a-124b that are associated with the alternate records 123a-123n represented in the vector. For example, a Vector of Alternates 122a stores alternate records 123a-123c. The alternate record 123a stores request and response information indicating that a stored object 124a associated with the alternate record is prepared in the English language. Another alternate record 123b stores information indicating that its associated stored object 124b is intended for use with the Microsoft Internet Explorer browser. The stored objects 124a, 124b referenced by the alternate records 123a, 123b are stored contiguously with the Vectors of alternates 122a-122n.

The Size value 120 within each alternate record indicates the total size in bytes of one of the associated Vectors of Alternates 122a-122n and the stored object 124. When the cache 80 references a Vector of Alternates 122a based on the disk location value 118, the cache reads the number of bytes indicated by the Size value. For example, in the case of the Vectors of Alternates shown in Figure 4B, the Size value would indicate the length of the Vector of Alternate 122a plus the length of its associated stored object 124a. Accordingly, by referencing the Size value, the cache 80 reads the vector as well as the stored object. In this way, the cache 80 "reads ahead" of the Vector of Alternates 122 and retrieves all of the objects 50 from the storage devices 90a-90n. As a result, both the Vector of Alternates and the objects 50 are read from the storage device using a single seek operation by the storage device. Consequently, when there is a hit in the cache 80, in the majority of cases (where there is a single alternate) the requested object 52 is retrieved from a storage device using a single seek.

When the disk location value 118 directly references a stored object 124, rather than a Vector of Alternates 122, the Size value 120 indicates the size of the object as stored in the disk block. This value is used to facilitate single-seek retrieval of objects, as explained further herein.

THE OPEN DIRECTORY

In one embodiment, the cache 80 further comprises an Open Directory 130. The Open Directory 130 stores a plurality of linked lists 132a-132n, which are themselves composed of a plurality of list entries 131a-131n. Each of the linked lists 132a-132n is associated with one of the sets 110a-110n in the Directory Table 110. The Open Directory 130 is stored in volatile main memory. Preferably, each list entry 131a-131n of the Open Directory 130 stores an object key that facilitates associative lookup of an information object. For example, each item within each linked list 132a-132n stores a complete object key 56 for an object 52.

The Open Directory accounts for objects that are currently undergoing transactions, to provide mutual exclusion against conflicting operations. For example, the Open Directory is useful in safeguarding against overwriting or deleting an object that is currently being read. The Open Directory also buffers changes to the Directory Table 110 before they are given permanent effect in the Directory Table 110. At an appropriate point, as discussed below, a synchronization operation is executed to move the changes reflected in the Open Directory 130 to the Directory Table 110. This prevents corruption of the Directory Table 110 in the event of an unexpected system failure or crash.

Further, in one embodiment, when an object is requested from the cache 80, the Open Directory 130 is consulted first; it is considered the most likely place to yield a hit, because it contains references to the most recently used information objects. The Open Directory in this form serves as a cache in main memory for popular data.

DISK DATA LAYOUT AND AGGREGATION

After the Open Directory 130, Tag Table 102 and Directory Table 110 have been accessed to determine the location of a stored object 124, the object must be read from storage and transmitted to the user that requested the object. To improve the efficiency of read operations that are used to retrieve objects 50 from the cache 80, certain data aggregation techniques are used when initially storing the data. When data is initially stored on disk according to the data aggregation techniques described herein, the efficiency of subsequent reads is improved greatly.

Figure 6 is a block diagram of a data storage arrangement for use with the cache 80 and the storage devices 90a-90n. A storage device 90a, such as a disk drive, stores data in plurality of

pools 200a-200n. A pool is a segment or chunk of contiguous disk space, preferably up to 4 Gbytes in size. Pools can be allocated from pieces of files, or segments of raw disk partitions.

Each pool, such as pool 200n, comprises a header 202 and a plurality of fixed size storage spaces referred to herein as "arenas" 204a through 204n. The size of the arenas is preferably configurable or changeable to enable optimization of performance of the cache 80. In the preferred embodiment, each of the arenas 204a-204n is a block approximately 512 Kbytes to 2 Mbytes in size.

Data to be written to arenas is staged or temporarily stored or staged in a "write aggregation buffer" in memory. This buffer accumulates data, and when full, the buffer is written contiguously, in one seek, to an arena on disk. The write aggregation buffer improves the performance of writes, and permits sector alignment of data, so data items can be directly read from raw disk devices.

The write aggregation buffer is large enough to hold the entire contents of an arena. Data is first staged and consolidated in the write aggregation buffer, before it is dropped into the (empty) arena on disk. The write aggregation buffer also contains a free top pointer that is used to allocate storage out of the aggregation buffer as it is filling, an identifier naming the arena it is covering, and a reference count for the number of active users of the arena.

Each pool header 202 stores a Magic number, a Version No. value, a No. of Arenas value, and one or more arena headers 206a-206n. The Magic number is used solely for internal consistency checks. The Version No. value stores a version number of the program or process that created the arenas 206a-206n in the pool. It is used for consistency checks to ensure that the currently executing version of the cache 80 can properly read and write the arenas. The No. of Arenas value stores a count of the number of arenas that are contained within the pool.

For each of the arenas in the pool, the pool header 202 stores information in one of the arena headers 206a-206n. Each arena header stores two one-bit values that indicate whether the corresponding arena is empty and whether the arena has become corrupted (e.g. due to physical disk surface damage, or application error).

As shown in Figure 6 in the exemplary case of an arena 204a, each arena comprises one or more data fragments 208a-208n. Each fragment 208a-208n comprises a fragment header 208d and fragment data 208e. The fragment data 208e is the actual data for an object that is stored in the cache 80. The data for an entire stored object may reside within a single fragment, or may be stored within multiple fragments that may reside in multiple arenas. The fragment header 208d stores a Magic number value 206c, a key value 206a and a length value 206b.

The length value 206b represents the length in bytes of the fragment, including both the fragment header 208d and the fragment data 208e. The key value 206a is a copy of the object

key, stored in its entirety, of the object whose data is in the fragment. Thus, the key value 206c can be used to look up the directory block that points to the first fragment that holds data of the object whose data is contained in the fragment.

According to one embodiment, the complete object key 56 is stored in association with the last fragment associated with a particular object. When an object 52 is stored in the cache 80 for the first time, the object key 56 is computed incrementally as object data is read from the originating server 40. Thus, the final value of the object key 56 cannot be known until the entire object 52 is read. The object key 56 is written at the end of the chain of fragments used to store the object, because the value of the key is not known until the last fragment is written, and because modifying existing data on disk is slow. In alternate embodiments, the fragment header can store other metadata that describes the fragment or object.

The write aggregation buffer contains a " free top pointer" 210 indicating the topmost free area of the buffer 204a. The top pointer 210 identifies the current boundary between used and available space within the buffer 204a. The top pointer 210 is stored to enable the cache 80 to determine where to write additional fragments in the buffer. Everything below (or, in Figure 6, to the left of) the top pointer 210 contains or has already been allocated to receive valid data. The area of the arena 204a above the top pointer 210 (to the right in Figure 6) is available for allocation for other information objects. Preferably, each fragment includes a maximum of 32 kilobytes of data. Fragments start and end on standard 512-byte boundaries of the storage device 90a. In the context of the World Wide Web, most objects are relatively small, generally less than 32K in size.

Each arena may have one of two states at a given time: the empty state or the occupied state. The current state of an arena is reflected by the Empty value stored in each arena header 206a-206n. In the occupied state, some portion of the arena is storing usable data. A list of all arenas that are currently empty or free is stored in memory. For example, main memory of the workstation that runs the cache 80 stores an array of pointers to empty arenas. In alternate embodiments, additional information can be stored in the header 206a-n of each arena. For example, the header may store values indicating the number of deleted information objects contained in the arena, and a timestamp indicating when garbage collection was carried out last on the arena.

Although three fragments are shown in Figure 6 as an example, in practice any number of fragments may be stored in an arena until the capacity of the arena is reached. In addition, the number of pools and the number of arenas shown in Figure 6 are merely exemplary, and any number may be used.

The above-described structure of the arenas facilitates certain consistent and secure mechanisms of updating data for objects that are stored in fragments of the arenas. Figure 7 is a block diagram relating to updating one of the arenas 204a-204n of Figure 6. Figure 7 shows an arena 204a containing a first information object 208b having a header 206 and data fragments 208a-208c. Top pointer 210 points to the topmost active portion of the arena 204a, which is the end of the data segment 208c. Preferably, the Directory Table is updated only after a complete information object has been written to an arena, including header and data, and only after the top pointer of the arena has been moved successfully. For example, a complete information object is written to the arena 204a above the top pointer 210, and the top pointer is moved to indicate the new top free location of the arena. Only then is the Directory Table updated.

The delayed updating of the Directory Table is carried out to ensure that the Directory Table remains accurate even if a catastrophic system failure occurs during one of the other steps. For example, if a disk drive or other element of the system crashes before completion of one of the steps, no adverse effect occurs. In such a case, the arena 204a will contain corrupt or incomplete data, but the cache 80 will effectively ignore such data because nothing in the Directory Table 110, indexes or hash tables is referencing the corrupt data. In addition, using the Garbage Collection process described herein, the corrupt or incomplete data is eventually reclaimed.

MULTI-FRAGMENT OBJECTS

In Figure 3, the directory table block 112b that is arrived at based on the object key of object 52 includes a pointer directly to the fragment in which the object 52 is stored. This assumes that object 52 has been stored in a single fragment.

However, large objects may not always fit into a single fragment, for two reasons. First, fragments have a fixed maximum size (preferred value is 32 KB). Objects greater than 32 KB will be fragmented. Second, the system must pre-reserve space in the write aggregation buffer for new objects. If the object store does not know the size of the incoming object, it may guess wrong. The server may also misrepresent the true (larger) size of the object. In both cases, the object store would create a chain of fragments to handle the overflow.

Therefore, a mechanism is provided for tracking which fragments contain data from objects that are split between fragments. Figure 5 is a block diagram of a preferred structure for keeping track of related fragments.

For the purpose of explanation, it shall be assumed that an object X is stored in three fragments 208a, 208b and 208c on storage devices 90a-90n. Using the object key for object X, the cache traverses the Tag Table to arrive at a particular block 141a within the Directory Table

110. Block 141a is the head of a chain of blocks that identify successive fragments that contain the object X. In the illustrated example, the chain is includes blocks 141a, 141b, 141c, 141d and 141e, in that order, and is formed by pointers 128a through 128d.

According to one embodiment, the head block 141a comprises a subkey value 126 and a block pointer 128a. Preferably, the subkey value 126 is 96 bits in length and comprises a subset of the value of the object key 56 for object X. The value of the block pointer 128a references the next block 141b in the chain.

Directory table block 141b comprises a fragment pointer 130a and a block pointer 128b. The fragment pointer 130a references a fragment 208a that stores the first portion of the data for the object X. The block pointer 128b of pointer block 141b references the next pointer block 141c in the chain. Like pointer block 141b, pointer block 141c has a fragment pointer 130b that references a fragment 208b. The block pointer 128c of pointer block 141c references the next pointer block 141d in the chain. Like pointer block 141c, pointer block 141d has a fragment pointer 130b that references a fragment 208c.

The object store needs a mechanism to chain fragments together. Traditional disk block chaining schemes require modifying pre-existing data on disk, to change the previous chain-link pointers to point the new next block values. Modification of pre-existing disk data is time-consuming and creates complexities relating to consistency in the face of unplanned process termination.

According to one embodiment of the invention, the need to patch new fragment pointers into extant fragments is removed by using "iterative functional pointers". Each fragment is assigned a key, and the key of the next fragment is assigned as a simple iterative function of the previous fragment's key. In this manner, fragments can be chained simply by defining the key of the next fragment, rather than by modifying the pointer of the previous fragment.

For example, the block pointer 128a is computed by applying a function to the value of subkey 126. The block pointer value 128b is computed by applying a function to the value of the block pointer 128a. The function used to compute the pointer values is not critical, and many different functions can be used. The function can be a simple accumulating function such that

$$key_n = key_{n-1} + 1$$

or the function can be a complex function such as the MD5 hash function

$$key_n = MD5(key_{n-1})$$

The only requirement is that the range of possible key values should be sufficiently large, and the iteration should be sufficiently selected, so that the chances of range collision or cyclic looping are small. In the very unlikely event of key collision, the object will be deleted from the cache.

The last pointer block 141d in the chain has a block pointer 128d that points to a tail block 141e. The tail block 141e comprises a reference to the first block 141a in the chain. According to one embodiment, the reference contained in the tail block 141e a 96-bit subkey 132 of the object key of object X. The cache can use the 96-bit subkey 132 to locate the head block 128a of the chain. The tail block 141e, and the looped pointer arrangement it provides, enables the cache 80 to locate all blocks in a chain, starting from any block in the chain.

Three fragments 208a, 208b, and 208c are shown in Figure 5 merely by way of example. In practice, an information object may occupy or reference any number of fragments, each of which would be identified by its own pointer block within the Directory Table 110.

When the object 52 is read from the storage device, the last fragment is read first to ensure that the content MD5 key stored there matches the directory key value. This test is done as a "sanity check" to ensure that the correct object has been located. If there is no match, a collision has occurred and an exception is raised.

SPACE ALLOCATION

Figure 10A is a flow diagram of a method of allocating space for objects newly entered into the cache and for writing such objects into the allocated space. The allocation and write method is generally indicated by reference numeral 640. Generally the steps shown in Figure 10A are carried out when a miss has occurred in the Directory Table and Tag Table, for example, at step 898 of Figure 8F.

Accordingly, in step 642, an information object that has been requested by a client, but not found in the cache, is looked up and retrieved from its original location. In a networked environment, the origin is a server 40, a cluster, or a disk. When the object is retrieved, in step 644 the method tests whether the object is of the type and size that can be stored in the cache, that is, whether it is "cacheable."

Examples of non-cacheable objects include Web pages that are dynamically generated by a server application, panes or portions of Web pages that are generated by client side applets, objects that are constructed based upon dynamic data taken from a database, and other non-static objects. Such objects cannot be stored in the cache because their form and contents changes each time that they are generated. If such objects were to be stored in the cache, they would be unreliable or incorrect in the event that underlying dynamic data were to change between cache accesses. The process determines whether the object is cacheable by examining information in the HTTP response from the server 40 or other source of the object.

If the object is cacheable, then in step 646 the method obtains the length of the object in bytes. For example, when the invention is applied to the World Wide Web context, the length of

a Web page can be included in metadata that is carried in an HTTP transaction. In such a case, the cache extracts the length of the information object from the response information in the HTTP message that contains the information object. If the length is not present, and estimate is generated. Estimates may be incorrect, and will lead to fragmented objects.

As shown in block 648, space is allocated in a memory-resident write aggregation buffer, and the object to be written is streamed into the allocated buffer location. In a preferred embodiment, block 648 involves allocating space in a write aggregation buffer that has sufficient space and is available to hold the object. In block 650, the cache tests whether the write aggregation buffer has remaining free space. If so, the allocation and write process is complete and the cache 80 can carry out other tasks. When the write aggregation buffer becomes full, then the test of block 650 is affirmative, and control is transferred to block 656.

In block 656, the cache writes the aggregation buffer to the arena it is shadowing. In step 660, the Directory is updated to reflect the location of the new information object.

The foregoing sequence of steps is ordered in a way that ensures the integrity of information objects that are written to the cache. For example, the Directory is updated only after a complete information object has been written to an arena, including header and data. For example, if a disk drive or other element of the system crashes before completion of step 652 or step 658, no adverse effect occurs. In such a case, the arena will contain corrupt or incomplete data, but the cache will effectively ignore such data because nothing in the indexes or hash tables is referencing the corrupt data. In addition, using the garbage collection process described herein, the corrupt or incomplete data is eventually reclaimed.

GARBAGE COLLECTION

Figure 8A is a flow diagram of a method of garbage collection that can be used with the cache 80. Figure 8B is a flow diagram of further steps in the method of Figure 8A, and will be discussed in conjunction with Figure 8A. Preferably, the garbage collection method is implemented as an independent process that runs in parallel with other processes that relate to the cache. This enables the garbage collection method to periodically clean up cache storage areas without interrupting or affecting the operation of the cache.

1. GENERAL PROCESS

In the preferred embodiment, "garbage collection" generally means a process of scanning target arenas, identifying active fragments or determining whether to delete fragments, writing the active fragments contiguously to new arenas, and updating the Directory Table to reference the new locations of the fragments. Thus, in a very broad sense the method is of the "evacuation" type, in which old or unnecessary fragments are deleted and active fragments are

written elsewhere, so that at the conclusion of garbage collection operations on a particular arena, the arena is empty. Preferably, both the target arenas and the new arenas are stored and manipulated in volatile memory. When garbage collection is complete, the changes carried out in garbage collection are written to corresponding arenas stored in non-volatile storage such as disk, in a process called synchronization.

In step 802, one of the pools 200a-200n is selected for garbage collection operations. Preferably, for each pool 200a-200n of a storage device 90a, the cache stores or can access a value indicating the amount of disk space in a pool that is currently storing active data. The cache also stores constant "low water mark" and "high water mark" values, as indicated by block 803. When the amount of active storage in a particular pool becomes greater than the "high water mark" value, garbage collection is initiated and carried out repeatedly until the amount of active storage in the pool falls below the "low water mark" value. The "low water mark" value is selected to be greater than zero, and the "high water mark" value is chosen to be approximately 20% less than the total storage capacity of the pool. In this way, garbage collection is carried out at a time before the pool overflows or the capacity of the storage device 90a is exceeded.

2. USAGE-AWARE GARBAGE COLLECTION

In step 804, one of the arenas is selected as a target for carrying out garbage collection. The arena is selected by a selection algorithm that considers various factors. As indicated by block 805, the factors include, for example, whether the arena is the last arena accessed by the cache 80, and the total number of accesses to the arena. In alternate embodiments, the factors may also include the number of information objects that have been deleted from each arena, how recently an arena has been used, how recently garbage collection was previously carried out on each arena, and whether an arena currently has read or write locks set on it. Once the arena is selected for garbage collection, all of the fragments inside the object are separately considered for garbage collection.

In step 806, one of the fragments within the selected arena is selected for garbage collection. In determining which fragment or fragments to select, the cache 80 takes into account several selection factors, as indicated by block 807. In the preferred embodiment, the factors include: the time of the last access to the fragment; the number of hits that have occurred to an object that has data in the fragment; the time required to download data from the fragment to a client; and the size of the object of which the fragment is a part. Other factors are considered in alternate embodiments. Values for these factors are stored in a block 112a-112n that is associated with the object for which the fragment stores data.

In block 808, the cache determines whether a fragment should be deleted. In the preferred embodiment, block 808 involves evaluation of certain performance factors and optimization considerations.

Caches are used for two primary, and potentially conflicting, reasons. The first reason is improving client performance. To improve client performance, it is desirable for a garbage collector to retain objects that minimize server download time. This tends to bias a garbage collector toward caching documents that have been received from slow external servers. The second reason is minimizing server network traffic. To minimize server traffic, it is desirable for a garbage collector to retain objects that are large. Often, these optimizations conflict.

By storing values that identify the time required to download an object, the size of the object, and the number of times the object was hit in cache, the garbage collector can estimate, for each object, how much server download time was avoided and how much server traffic was disabled, by serving the cached copy as opposed to fetching from the original server. This metric measures the inherent "value" of the cached object.

The cache administrator then configures a parameter between 0 and 1, indicating the degree to which the cache should optimize for time savings or for traffic savings. The foregoing values are evaluated with respect to other objects in the arena, with respect to the amount of space the object is consuming, and with respect to objects recently subjected to garbage collection. Based on such evaluation, the cache 80 determines whether to delete the fragment, as shown in step 808.

If the fragment is to be deleted, then in step 812 it is deleted from the arena by marking it as deleted and overwriting the data in the fragment. When an object 52 is stored in multiple fragments, and the garbage collection process determines that one of the fragments is to be deleted, then the process deletes all fragments associated with the object. This may involve following a chain of fragments, of the type shown in Figure 5, to another arena or even another pool.

If the fragment is not to be deleted, then in step 810 the fragment is written to a new arena. Figure 8B, which is discussed below, shows preferred sub-steps involved in carrying out step 810.

After the fragment is deleted or moved to another arena, in step 814 the Directory Table 110 is updated to reflect the new location of the fragment. Step 814 involves using the value of the key 206a in the fragment header 208d associated with a fragment 208n to be updated to look up a block 112a-112n that is associated with the fragment. When the correct Directory Table block 112a-112n is identified, the disk location value 118 in the block is updated to reflect the

new location of the fragment. If the fragment has been deleted, then any corresponding Directory Table entries are deleted.

Step 816 indicates that the method is complete after the Directory Table 110 is updated. However, it should be understood that the steps of Figure 8A are carried out for all pools, all arenas within each pool, and all fragments within each arena.

3. WRITING FRAGMENTS TO NEW ARENAS

Figure 8B is a flow diagram of steps involved in carrying out step 810, namely, writing a fragment that is to be preserved to a new arena. The process of writing evacuated fragments to new arenas is completely analogous to writing original fragments. The data is written into a write aggregation buffer, and dropped to disk arenas when full.

In step 590, the directory tables are updated to reflect the change in location of the fragment. In the preferred embodiment, step 590 involves writing update information in the Open Directory 130 rather than directly into the Directory Table 110. At a later time, when the process can verify that the fragment data 208e has been successfully written to one of the storage devices 90a-90n, then the changes reflected in the Open Directory 130 are written into or synchronized with the Directory Table 110.

This process is used to ensure that the integrity of the Directory Table 110 is always preserved. As noted above, buffered storage is used for the fragments; thus, when a fragment is updated or a new fragment is written, the fragment data is written to a buffer and then committed to a disk or other storage device at a future time. Thus, during garbage collection, it is possible that a fragment that has been moved to a new arena is not actually written on one of the storage devices when the garbage collection process is ready to update the Directory Table. Therefore, information about the change is stored in the Open Directory 130 until the change is committed to disk.

In step 592, the original arena is examined to test whether it has other fragments that might need to be reclaimed or moved to a new arena. If other objects are present, then control returns to step 806 of Figure 8A, so that the next object can be processed. If no other objects are present in the current arena, then in step 594, the top pointer of the current arena is reset.

4. BUFFERING

In the preferred embodiment, read and write operations carried out by the cache 80 and the garbage collection process are buffered in two ways.

First, communications between the cache 80 and a client 10a that is requesting an object from the browser are buffered through a flow-controlling, streaming, buffering data structure called a VConnection. In the preferred embodiment, the cache 80 is implemented in a set of computer programs prepared in an object-oriented programming language. In this embodiment,

the VConnection is an object declared by one of the programs, and the VConnection encapsulates a buffer in memory. Preferably, the buffer is a FIFO buffer that is 32 Kbytes in size.

When a client 10a-10c connects to the cache 80, the cache assigns the client to a VConnection. Data received from the client 10a is passed to the cache 80 through the VConnection, and when the cache needs to send information to the client 10a, the cache writes the information to the VConnection. The VConnection regulates the flow of data from the cache 80 to match the data transmission speed used by the client 10a to communicate with the cache. In this way, use of the VConnection avoids an unnecessary waste of main memory storage. Such waste would arise if an object being sent to the client 10a was copied to memory in its entirety, and then sent to the client; during transmission to a slow client, main memory would be tied up unnecessarily. Buffered I/O using these mechanisms tends to reduce the number of sequential read and write operations that are carried out on a disk.

5. SYNCHRONIZATION AND CONSISTENCY ENFORCEMENT

Regularly during the garbage collection process and during operation of the cache 80, a synchronization process is carried out. The synchronization process commits changes reflected in the Open Directory 130 to the Directory Table 110 and to stable storage, such as non-volatile storage in one or more of the storage devices 90a-90n. The goal is to maintain the consistency of the data on disk at all times. That is, at any given instant the state of the data structures on disk is 100% consistent and the cache can start up without requiring checking. This is accomplished through careful ordering of the writing and synchronization of data and meta-data to the disk.

For the purposes of discussion, in this section, 'data' refers to the actual objects the cache is being asked to store. For instance, if the cache is storing an HTML document, the data is the document itself. 'Meta-data' refers to the additional information the cache needs to store in order to index the 'data' so that it can be found during a subsequent lookup() operation as well as the information it needs to allocate space for the 'data'. The 'meta-data' is comprises the directory and the pool headers. The directory is the index the cache uses for associating a key (a name) with a particular location on disk (the data). The cache uses the pool headers to keep track of what disk space has been allocated within the cache.

The cache uses two rules to maintain the consistency of the data structures on disk. The first rule is that meta-data is always written down after the data it points to. The rationale for the first rule is that the cache has no "permanent" knowledge of an object being in the cache until the meta-data is written. If the cache were to write down the meta-data before the data and then crash, the meta-data would associate an object name with invalid object data on disk. This is undesirable, since the cache would then have to use heuristics to try and determine which meta-data points to good data and which points to bad.

The second rule is that a pool arena cannot be marked as empty in the pool header until all the directory meta-data that points to the arena has been deleted and written to disk. This is necessary so that a crash cannot cause an empty arena to exist for which directory meta-data points to it. The problem this can cause is that the empty arena can become filled with new data, since it is empty and therefore it is available for new data to be written into it. However, "old" directory meta-data points to the same location as the new data. It is possible for accesses to the old directory meta-data to return the new data instead of either returning the old data or failing.

Figure 8C is a flow diagram of a preferred synchronization method 820 that implements the foregoing two rules. In block 822, an object is written to the cache. Block 822 involves the steps of block 824 and block 826, namely, creating metadata in the Open Directory, and writing and syncing the object data to disk.

The steps of blocks 828 through 820' are carried out periodically. As indicated in block 828, for each piece of meta-data in the open directory table, a determination is made whether the data that the metadata points to is already synchronized to disk, as shown in block 821. If so, then in block 823, the cache copies the metadata that points to the stable data from the Open Directory to the Directory Table. In block 825, the changes are synchronized to disk.

In block 827, garbage collection is carried out on an arena. Block 827 may involve the steps shown in Figure 8A. Alternatively, garbage collection generally involves the steps shown in block 829, block 831, and block 820'. As shown in block 829, for each fragment in the arena, the cache deletes the directory metadata that points to the segment, and writes the directory metadata to disk. In block 831, the pool header is modified in memory such that the arena is marked as empty. In block 820', the pool header is written and synced to disk.

The steps that involve writing information to disk preferably use a "flush" operation provided in the operating system of the workstation that is running the cache 80. The "flush" operation writes any data in the buffers that are used to store object data to a non-volatile storage device 90a-90c.

Using the foregoing methods, the Directory Table is not updated with the changes in the Open Directory until the data that the changes describe is actually written to disk or other non-volatile storage. Also, the cache 80 postpones updating the arenas on disk until the changes undertaken by the garbage collection process are committed to disk. This ensures that the arenas continue to store valid data in the event that a system crash occurs before the Directory Table is updated from the Open Directory.

6. RE-VALIDATION

In the preferred embodiment, the cache provides a way to re-validate old information objects in the cache so that they are not destroyed in the garbage collection process.

Figure 12 is a flow diagram of a preferred re-validation process. In block 1202, an external program or process delivers a request to the cache that asks whether a particular information object has been loaded by a client recently. In response to the request, as shown in block 1204, the cache locates the information object in the cache. In block 1206, the cache reads a Read Counter value associated in the directory tables with the information object. In block 1208, the cache tests whether the Read Counter value is high.

If the Read Counter value is high, then the information object has been loaded recently. In that case, in block 1210 the cache sends a positive response message to the requesting process. Otherwise, as indicated in block 1212, the information object has not been loaded recently. Accordingly, as shown in block 1214, the cache sends a negative responsive message to the calling program or process. In block 1216, the cache updates an expiration date value stored in association with the information object to reflect the current date or time. By updating the expiration date, the cache ensures that the garbage collection process will not delete the object, because after the update it is not considered old. In this way, an old object is refreshed in the cache without retrieving the object from its origin, writing it in the cache, and deleting a stale copy of the object.

SCALED COUNTER UPDATING

Figure 10B is a flow diagram of a method of scaled counter updating. In the preferred embodiment, the method of Figure 10B is used to manage the Read Counter values that are stored in each block 112a-112n of a set of the Directory Table, as shown in Figure 3A. However, the method of Figure 10B is not limited to that context. The method of Figure 10B is applicable to any application that involves management of each of a plurality of objects that has a counter, and in which it is desirable to track the most recently used or least recently used objects. A key advantage of the method of Figure 10B in comparison to past approaches is that it enables large counter values to be tracked in a small storage area.

In the preferred embodiment, each of the Read Counter values stored in blocks 112a-112n is stored in three bit quantities. During operation of the cache 80, when a block is accessed, the Read Counter value of the block is incremented by one. The highest decimal number that can be represented by a three-bit quantity is 7. Accordingly, a Read Counter could overflow after being incremented seven times. To prevent counter overflow, while enabling the counters to track an unlimited number of operations that increment them, the method of Figure 10B is periodically executed.

The following discussion of the steps of Figure 10B will be more clearly understood with reference to Table 1:

TABLE 1 – SUCCESSIVE COUNTER VALUES

EVENT	COUNTERS		
	A	B	C
1: Start	1	1	1
2: Increment	2	1	1
3: Increment	7	3	1
4: Decrement	6	2	0
5: Reclaim	6	2	--

In Table 1, the EVENT column identifies successive events affecting a set of counter values, and briefly indicates the nature of the event. The COUNTERS heading indicates three counter values A, B, and C represented in separate columns. Each of the counter values A, B, C corresponds to a counter value that is stored in a different block 112a-112n of the Directory Index 110. Thus, each row of Table 1 indicates the contents of three counter values at successive snapshots in time.

Event 1 of Table 1 represents an arbitrary starting point in time, in which the hash table entries containing the counter values A, B, C each have been accessed once. Accordingly, the value of each counter A, B, C is one. At event 2, the cache has accessed the hash table entry that stores counter value A. Accordingly, counter A has been incremented and its value is 2; the other counters B, C are unchanged. Assume that several other hash table entry accesses then occur, each of which causes one of counters A, B, or C to be incremented. Thereafter, at event 3, the values of the counters A, B, C are 7, 3, and 1 respectively. Thus, counter A is storing the maximum value it can represent, binary 111 or decimal 7, and will overflow if an attempt is made to increment it to a value greater than 7.

At this point, the method of Figure 10B is applied to the counters A, B, C. In step 622, the value of all the counters is read. In step 624, the sum of all the counter values is taken. In the case of Table 1, the sum is given by 7+3+1 = 11. In step 626, the maximum sum that can be represented by all the counters is computed based upon the length in bits of the counter values. In the case of a three-bit value, the maximum value of one counter is 7 and the maximum value for the sum of three three-bit counters is 7x3=21. Alternatively, step 626 can be omitted; the maximum value can be stored as a constant that is available to the scaled counter method 620 and simply retrieved when needed.

In step 628, the method computes the value (maximum_value/2), truncating any remainder or decimal portion, and compares it to the sum of all the counters. In the example above, the relationship is

Sum = 11

Maximum_Value = 21

Maximum_Value / 2 = 10

(Sum > Maximum_Value / 2) = TRUE

Since the result is true, control is transferred to step 630, in which all the counter values are decremented by 1. The state of counters A, B, C after this step is shown by Event 4, "Decrement." Note that counter C, which represents the least recently used hash table entry, has been decremented to zero. At this point, least recently used hash table entries can be reclaimed or eliminated by scanning the corresponding counter values and searching for zero values. The result of this step is indicated in Event 5 of Table 1, "Reclaim." The values of counters A and B are unchanged, and the value of counter C is undefined because its corresponding hash table entry has been deleted from the hash table.

When the method of Figure 10B is repeated periodically and regularly, none of the plurality of counter values will overflow. Also, least recently used entries are rapidly identified by a counter value of zero, and can be easily eliminated from the cache. Counter values can be maintained in few bits even when hash table entries are accessed millions of times. Thus, the method of Figure 10B provides a fast, efficient way to eliminate least recently used entries from a list.

CACHE OPERATIONS

In the preferred embodiment, the cache 80 is implemented in one or more computer programs that are accessible to external programs through an API that supports read and write operations. The read and write operations are carried out on the Open Directory 130, which is the only structure of the cache 80 that is "visible" to external programs or processes. The read operation is invoked by an external program that wants to locate an object in the cache. The write operation is invoked by a program that wants to store an object in the cache. Within the programs that make up the cache 80, operations called lookup, remove, checkout, and checkin are supported. The lookup operation looks up an object in the Open Directory based upon a key. The remove operation removes an object from the Open Directory based upon a key. The checkout operation obtains a copy of a block from the Directory Table 110 in an orderly manner so as to ensure data consistency. The checkin operation returns a copy of a block (which may have been

modified in other operations) to the Directory Table 110. In other embodiments, a single cache lookup operation combines aspects of these operations.

1. LOOKUP

In an alternate embodiment, a LOOKUP operation is used to determine whether a particular object identified by a particular name is currently stored in the cache 80. Figure 9A is a flow diagram of steps carried out in one embodiment of the LOOKUP operation, which is generally designated by reference numeral 902. The LOOKUP operation is initiated by a command from the protocol engine 70 to the cache 80 when a request message from a client 10a seeks to retrieve a particular object from the server 40. The request message from the client 10a identifies the requested object by its name.

When the process is applied in the context of the World Wide Web, the name is a Uniform Resource Locator (URL). In step 904, the cache 80 converts the name of the object to a key value. In the preferred embodiment, the conversion step is carried out as shown in Figure 3B. The object name 53 or URL is passed to a hash function, such as the MD5 one-way has function. The output of the hash function is an object name key 62. The object name key 62 can be broken up into one or more subkey values 64, 66.

In step 906, the cache 80 looks up the request key value in the Open Directory 130. The Open Directory is consulted first because it is expected to store the most recently requested objects and therefore is likely to contain the object in the client request. Preferably, step 906 involves using one of the subkey values as a lookup key. For example, a 17-bit or 18-bit subkey value can be used for the lookup.

In step 908, the cache 80 tests whether the subkey value has been found in the Open Directory. If the subkey value has been found in the Open Directory, then in step 910 the cache 80 retrieves the object from one of the storage devices, and delivers the object to the client. The retrieval sub-step involves the sub-steps described above in connection with locating objects in pools, arenas, and fragments of non-volatile storage in the storage devices 90a-90c. The delivery sub-step involves constructing an HTTP response to the client that includes data of the object, opening an HTTP connection to the client, and sending the HTTP request to the client.

If the subkey value is not found in the Open Directory, then in step 912, the cache 80 looks up the request subkey value in the Tag Table 102. In step 914, the cache 80 tests whether the subkey value was found in the Tag Table 102. If no match was found, then in step 916 the cache 80 stores information about the fact that no match occurred, for later use as described below. The information can be a bit indicating that a miss in the Tag Table 102 occurred.

In step 918, the cache 80 looks up the subkey value in the Directory Table. If the test of step 914 was affirmative, then the cache 80 retrieves a subkey value matching the request subkey

value from one of the entries 106 of the tag Table 102. Its value is used as a key to look up the request key value in the Directory Table. In step 920, the cache 80 tests whether the request key value was found in the Directory Table. If a hit occurs, and there was a miss in the Tag Table as indicated by the information stored in step 916, then in step 922 the cache 80 updates the Open Directory with information related to the Directory Table hit. Control is then passed to step 910 in which the object is obtained and delivered to the client in the manner described above.

If the test of step 920 is negative, then the requested object is not in the cache, and a cache miss condition occurs, as indicated in step 924. In response to the miss condition, in step 926 the cache 80 obtains a copy of the requested object from the server that is its source. For example, in the Web context, the cache 80 opens an HTTP connection to the URL provided in the client's request, and downloads the object. The object is then provided to the client and stored in the cache for future reference.

In a preferred embodiment, the LOOKUP operation is implemented as a method of an object in an object-oriented programming language that receives a key value as a parameter.

2. CACHE OPEN READ PROCESS

Figure 9E is a flow diagram of a preferred process of reading an object that is identified by an object name (such as a URL) from the cache. In the preferred embodiment, the process of Figure 9E is called "open_read," and represents the sole external interface of the cache 80. It is advantageous, to ensure control and consistency of data in the cache, to enable external programs to access only operations that use or modify the Open Directory 130. Preferably, the process of Figure 9E is implemented as a program or programmatic object that receives an object name, and information about the user's particular request, as input parameters. The read process returns a copy of an object associated with a key that is found in the cache using the lookup process. Thus, the read process, and other processes that are invoked or called by it, are an alternative to the LOOKUP operation described above in connection with Figure 9A.

In step 964, the process checks out a Vector of Alternates so that alternates in the vector can be read. Preferably, step 964 involves invoking the checkout_read process described herein in connection with Figure 8D, providing a key derived from the object name as a parameter. Checking out a vector involves checking out a block from the Open Directory that has a pointer to the vector, and reaching the block from the cache.

If the checkout operation is successful, then in step 966 the process uses the request information to select one of the alternates from among the alternates in the vector. This selection is carried out in the manner described above in connection with the Vector of Alternates 122. In an embodiment, the selection operation is carried out by another program or programmatic object that returns a success/failure indication depending upon whether a suitable alternate is located. If

the selection is successful, then in step 968 the process checks the Vector of Alternates back in. In step 970, the process reads the object that is pointed to by the selected alternate.

If step 964 or step 966 results in failure, then the requested document does not exist in the cache. Accordingly, in step 972 the process returns a "no document" error message to the calling program or process.

3. CACHE OPEN WRITE PROCESS

Figure 9F is a flow diagram of a process of writing an object into the cache. As in the case of the read process described above in connection with Figure 9E, the write process preferably is implemented as an "open_write" method that is the sole interface of the cache 80 to external programs needing to store objects in the cache. Preferably, the process of Figure 9F is implemented as a program or method that receives an object name, request information, and response information as input parameters. The object name identifies an object to be written into the cache; in the preferred embodiment, the object name is a name key 62 derived from a URL using the mechanism shown in Figure 3B.

The write process is initiated when a client 10a has requested an object 52 from the cache 80 that is not found in the cache. As a result, the cache 80 opens an HTTP transaction with the server 40 that stores the object, and obtains a copy of the object from it. The request information that is provided to the cache write process is derived from the HTTP request that came from the client. The response information is derived from the response of the server 40 to the cache 80 that supplies the copy of the object.

In step 974, the process checks out a Vector of Alternates. This step involves computing a key value based upon the object name, looking up a set and a block in the Open Directory that map to the key value, and locating a Vector of Alternates, if any, that corresponds to the block. If no vector exists, as shown in step 984, a new vector is created

If a vector is successfully checked out or created, then in step 976 the process uses the request information to define a new alternate record 123a-123n within the current alternate. The new alternate record references the location of the object, and contains a copy of the request information and the response information. The new alternate is added to the Vector of Alternates. Duplicate alternate records are permitted; the Vector of Alternates can contain more than one alternate record that contains the same request and response information. Testing existing alternate records to identify duplicates is considered unnecessary because only a small incremental amount of storage is occupied by duplicate alternate records.

In step 978, the modified vector is checked into the cache using the steps described above. In step 980, the object is written to one of the data storage devices 90a-90c in the manner

described above, using the key value. If the key is found to be in use during step 980, then the write operation fails. This avoids overwriting an object identified by a key that is being updated.

4. CACHE UPDATE PROCESS

Figure 9G is a flow diagram of a cache update process. The update process is used to modify a Vector of Alternates to store different request information or response information. Generally, the update process is invoked by the protocol engine 70 when the cache 80 is currently storing an object 52 that matches a request from a client 10a, but the protocol engine determines that the object has expired or is no longer valid. Under these circumstances, the protocol engine 70 opens an HTTP transaction to the server 40 that provided the original object 52, and sends a message that asks the server whether the object has changed on the server. This process is called "revalidation" of the object 52. If the server 40 responds in the negative, the server will provide a short HTTP message with a header indicating that no change has occurred, and providing new response information. In that case, the protocol engine 70 invokes the cache update process in order to move the new response information about the object 52 into the cache 80.

If the server 40 responds affirmatively that the object 52 has changed since its expiration date or time in the cache 80, then the update process is not invoked. Instead, the server 40 returns a copy of the updated object 52 along with a new expiration date and other response information. In that case, the protocol engine 70 invokes the cache write process and the create processes described above to add the new object 52 to the cache 80.

As shown in Figure 9G, the update process receives input parameters including an object name, an "old" identifier, request information, and response information. The object name is a URL or a key derived from a URL. The request information and response information are derived from the client's HTTP request for the object 52 from the cache 80, and from the response of the server 40 when the cache obtains an updated copy of the object from the cache.

The "old" identifier is a value that uniquely identifies a pair of request information and response information. In the preferred embodiment, when a cache miss causes the cache 80 to write a new object into the cache, information from the client request is paired with response information from the server that provides a copy of the object. Each pair is given a unique identifier value.

In step 986, the process checks out a Vector of Alternates corresponding to the object name from the cache. Preferably, this is accomplished by invoking the checkout_write process described herein. This involves using the object name or URL to look up an object in the Open Directory, the Tag Table, and the Directory Index, so that a corresponding Vector of Alternates is

obtained. If the checkout step fails, then in step 996 the process returns an appropriate error message.

If the checkout is successful, then in step 988 a copy or clone of the vector is created in main memory. A request/response identifier value is located within the vector by matching it to the Old Identifier value received as input to the process. The old identifier value is removed and a new identifier is written in its place. The new identifier uniquely identifies the new request and response information that is provided to the process as input.

In step 990, the new vector is written to one of the storage devices 90a-90c, and in step 992 the new vector is checked in to the cache. In carrying out these steps, it is desirable to completely write the clone vector to the storage device before the vector is checked in. This ensures that the writing operation is successful before the directory tables are modified to reference the clone vector. It also ensures that the old vector is available to any process or program that needs to access it.

5. DIRECTORY LOOKUP

Figure 9C is a flow diagram of a preferred embodiment of a process of looking up information in the Open Directory 130. The process of Figure 9C is implemented as a program process or method that receives a subkey portion of a name key 62 as an input parameter. In preceding steps that are not shown, it will be understood that the protocol engine 70 receives an object name, such as a URL. For example, a URL is provided in an HTTP request issued by a client to a server that is operating the cache. The protocol engine 70 applies a hash function to the object name. The hash function yields, as its result or output, a name key that identifies a set in the cache.

In step 948, the process attempts to check out one or more blocks that are identified by the subkey from the Directory Index. The block checkout step preferably involves invoking the checkout_read process described herein. Thus,

If the checkout attempt results in a failure state, then in step 950 the process returns an error message to the program or process that called it, indicating that a block matching the input subkey was not found in the cache. Control is passed to step 952 in which the process concludes.

If the checkout attempt is successful, then a copy of a block becomes available for use by the calling program. In step 954, the block that was checked out is checked in again. In step 956, the process returns a message to the calling program indicating that the requested block was found. Processing concludes at step 952.

Thus, a cache search operation involves calling more primitive processes that seek to check out a block identified by a key from the Open Directory. If the primitives do not find the block in the Open Directory, the Directory Index is searched.

When a block is found, it is delivered to the client. For example, when the invention is applied to the World Wide Web context, the data block is delivered by opening an HTTP connection to the client and transmitting the data block to the client using an HTTP transaction. This step may involve buffering several data blocks before the transaction is opened.

6. CACHE REMOVE PROCESS

Figure 9D is a flow diagram of a process of removing a block relating to an object from the cache. As in the case of the checkout operations, the cache remove process receives a key value as input. The process comprises steps 958 to 962. These steps carry out operations that are substantially similar to the operations of steps 948, 954, and 952 of Figure 9C. To accomplish removal of a block found in the cache, however, in step 960 the process sets the deletion flag, and checks the block in with the deletion flag set. As described herein in connection with the check-in process (steps 938 and 944 of Figure 9B), when the deletion flag is set, the block will be marked as deleted. Thereafter, the block is eventually removed from the Directory Index when the changes reflected in the Open Directory are synchronized to the Directory Index.

7. CHECKOUT READ OPERATION

Figure 8D is a flow diagram of a checkout_read operation that is used in connection with the Directory Table 110. The checkout_read operation is used to obtain a copy of a block from the Directory Table 110 that matches a particular key. Once the block is checked out from the Directory Table 110, the block can be read and used by the process that checked it out, but by no other process. Thereafter, to make the block available to other processes, the block is checked back in. Complementary checkout check-in processes are used in order to ensure that only one process at a time can modify a Directory Table block, a mechanism that is essential to ensure that the Directory Table always stores accurate information about objects in the cache. Thus, it will be apparent that the checkout and check-in processes is a primitive process that assists in searching the cache for a particular object.

As indicated in Figure 8D, the checkout_read operation receives a key value as input. In the preferred embodiment, the input key value is a subkey portion of a name key 62 that corresponds to an object name.

Because the object store will be modifying portions of memory and disk data structures, it needs to guarantee a brief period of mutual exclusion to a subset of the cache data structures in order to achieve consistent results. The cache data structures are partitioned into 256 virtual "slices", selected by 8 bits of the key. Each slice has an associate mutex lock. In step 832, the

process seeks to obtain the lock for the input key. If a lock cannot be obtained, the process waits the brief time until it becomes available. A lock can be unavailable if another transaction is modifying the small about of memory state associated with a key that falls in the same slice.

When a lock is obtained, the input key becomes unavailable for use by other processes. In step 834, the process determines which set 110a-110n of the Directory Table 110 corresponds to the key. The process then locates one of the block lists 132a, 132b of the Open Directory 130 that corresponds to the set of the Directory Table 110, by associating the value of a subkey of the input key with one of the block lists. In step 836, the process scans the blocks in the selected block list of the Open Directory 130, seeking a match of the input key to a key stored in one of the blocks.

If a match is found, then in step 838 the process tests whether the matching block is currently in the process of being created or destroyed by another process. If the matching block is currently in the process of being created or destroyed, then in step 840 an error message is returned to the protocol engine 70 indicating that the current block is not available.

On the other hand, if the matching block is not currently in the process of being created or destroyed, then the block can be used. Accordingly, in step 842 the process increments a read counter. The read counter is an internal variable, associated with the block, that indicates the number of processes or instances of programmatic objects that are reading the block. Such processes or objects are called "readers." In step 844, the process obtains a copy of the block, and returns it to the calling program or process.

If a match is not found in the scan of step 836, then in step 846, the process invokes a search of the Directory Table, seeking a match of the key to a set and block of the Directory Table using a process that is described further herein. If no match of the key is found in the search, then in step 848 the process returns an error message to the calling program or process, indicating that the requested object does not exist in the cache. Although the specific response to such a message is determined by the calling program or process, in the World Wide Web context, generally the proxy 30 contacts the server 40 that stores the object using an HTTP request, and obtains a copy of the requested object.

If a match is found during the Directory Index lookup of step 846, then in step 850 a corresponding block is added to the Open Directory. This is carried out by creating a new Open Directory block in main memory; initializing the block by copying information from the corresponding Directory Index block; and adding a reference to the new block to the corresponding list of blocks 132a, 132b.

8. CHECKOUT WRITE OPERATION

Figure 8E is a flow diagram of a checkout_write process or operation that is used in connection with the Open Directory 130. The checkout_write operation is used to obtain a copy of a block from the Open Directory 130 that matches a key that is passed to the process, for the purpose of modifying or updating the contents of the block, or an object or vector that is associated with the block. Once a block is checked out of the Open Directory 130 using checkout_write, other processes can modify the block or its associated object or vector. The block is then checked back in using the checkin process described herein. Using these operations, changes are stored in the Open Directory and then propagated to the Directory Table in an orderly manner.

As indicated in Figure 8E, the checkout_write process receives a key value as input. In the preferred embodiment, the input key value is a subkey portion of a name key 62 that corresponds to an object name. In step 854, the process seeks to obtain a lock on the designated key. If a lock cannot be obtained, the process waits until one is available.

When a lock is obtained, the key becomes unavailable for use by other processes. In step 856, the process determines which set 110a-110n of the Directory Table 110 corresponds to the key. The process then locates one of the block lists 132a, 132b of the Open Directory 130 that corresponds to the set of the Directory Table 110. In step 858, the process scans the blocks in the selected block list of the Open Directory 130, seeking a match of the input key to a key stored in one of the blocks.

If a match is found, then in step 864 the process tests whether the matching block is currently in the process of being created or destroyed by another process. If so, then in step 866 an error message is returned to the protocol engine 70 or cache 80 indicating that the current block is not available. If the matching block is not currently in the process of being created or destroyed, then the block can be used. Accordingly, in step 868 the process increments a write counter. The write counter is an internal variable, stored in association with the block, that indicates the number of processes or programmatic objects that are writing the block. In step 870, the process obtains a copy of the block, returns it to the calling program or process, and also marks the copy as being modified. The marking ensures that any changes made to the block will be reflected in the Directory Index when the Open Directory is synchronized to the Directory Index.

If a match is not found in the scan of step 858, then in step 860, the process invokes a search of the Directory Index using a process that is described further herein. If no match is found in the search, then in step 862 the process returns an error message to the calling program or process, indicating that the requested object does not exist in the cache. In the World Wide

Web context, typically the calling program would contact the originating server that stores the object using an HTTP request, and obtain a copy of the requested object.

If a match is found during the Directory Index lookup of step 860, then in step 874 a corresponding block is added to the Open Directory. This is carried out by creating a new Open Directory block in main memory; initializing the block by copying information from the corresponding Directory Index block; and adding a reference to the new block to the corresponding list of blocks 132a, 132b. Control is then passed to step 868, in which the write count is incremented and the process continues as described above in connection with steps 868-870.

9. CHECKOUT CREATE OPERATION

Figure 8F is a flow diagram of a checkout_create operation that is supported for use in connection with the Open Directory 130. The checkout_create operation is used to create a new block in the Open Directory 130 for a name key that corresponds to a new object that is being added to the cache. Once the block is created in the Open Directory 130, the object can be obtained by users from the cache through the Open Directory 130.

As indicated in Figure 8F, the checkout_create process receives a key value as input. In the preferred embodiment, the input key value is a subkey portion of a name key 62 that corresponds to an object name. In step 876, the process seeks to obtain a lock on the designated key. If a lock cannot be obtained, the process waits until one is available.

When a lock is obtained, the key becomes unavailable for use by other processes. In step 878, the process determines which set 110a-110n of the Directory Table 110 corresponds to the key. The process then locates the set of the Open Directory 130 that corresponds to the set of the Directory Table 110, using the set subkey bits of the input key. In step 880, the process scans the blocks in the selected block list of the Open Directory 130, seeking a match of the input key to a key stored in one of the blocks.

If a match is found, then an attempt is being made to create a block that already exists. Accordingly, in step 882 the process tests whether the matching block has been marked as deleted, and currently has no other processes reading it or writing it. If the values of both the reader counter and the writer counter are zero, then the block has no other processes reading it or writing it. If the values of either the reader counter or the writer counter are nonzero, or if the matching block has not been marked as deleted, then the block is a valid previously existing block that cannot be created. In step 884 an error message is returned to the protocol engine 70 or cache 80 indicating that the current block is not available to be created.

If the matching block is deleted and has no writers or readers accessing it, then the process can effectively create a new block by clearing and initializing the matching, previously

created block. Accordingly, in step 886 the process clears the matching block. In step 888 the process initializes the cleared block by zeroing out particular fields and setting the block's key value to the key. In block 890, the process increments the writer counter associated with the block, and marks the block as created. In step 892, the process returns a copy of the block to the calling process or programmatic object, and marks the block as being modified.

If a match is not found in the scan of step 880, then no matching block currently exists in the Open Directory 130. In step 894, the process carries out a search of the Directory Index using a process that is described further herein. If a match occurs, then in step 896, the process returns an error message to the calling program or process, indicating that the block to be created already exists in the cache and cannot be deleted.

If no match is found in the search, then no matching block currently exists in the entire cache. In step 898, the process creates a new Open Directory block, and adds a reference to that block to the list 132a, 132b associated with the set value computed in step 878. Control is passed to step 890, in which the processing continues as described above in connection with steps 890-892.

10. CHECKIN PROCESS

Figure 9B is a flow diagram of a block check-in process. The cache 80 carries out the process of Figure 9B to check a block into the Open Directory 130 after the block is read, modified, or deleted. In an embodiment, the process of Figure 9B is implemented as a program process or object that receives an identifier of a block as a parameter. Because the key is present in the checked out block, we do not need to pass in the key as an argument.

In step 930, the process attempts to get a lock for the key associated with the block. If no lock is available, then the process enters a wait loop until a lock is available. When a lock is available, in step 932 the process tests whether the block is being checked in after the block has been modified. If so, then in step 934 the writer count for the block is decremented, indicating that a process has completed writing the block.

In step 936, the process tests whether the check-in process has been carried out successfully. If this test is affirmative, then in step 942 the process copies the information in the current block to the corresponding original block in the Open Directory. In this way, the Open Directory is updated with any changes that were carried out by the process that modified the copy of the block that was obtained in the checkout process. Thereafter, and if the test of step 936 is negative, the process tests whether a delete check-in flag is set. The delete check-in flag indicates that the block is to be deleted after check-in. The delete flag is an argument to the checkin operation. If the flag is set, then in step 944 the process marks the block as deleted. Processing concludes at step 940.

If the test of step 932 is negative, then the block is not being modified. As a result, the only other possible state is that the block has been read. Accordingly, in step 946, the reader count is decremented.

IMPLEMENTATION OF METHODS

In the preferred embodiment, the methods described herein are carried out using a general-purpose programmable digital computer system of the type illustrated in Figure 11. Each of the methods can be implemented in several different ways. For example, the methods can be implemented in the form of procedural computer programs, object-oriented programs, processes, applets, etc., in either a single-process or multi-threaded, multi-processing system.

In a preferred embodiment, each of the processes is independent and re-entrant, so that each process can be instantiated multiple times when the cache is in operation. For example, the garbage collection process runs concurrently with and independent of the allocation and writing processes.

HARDWARE OVERVIEW

Figure 11 is a block diagram that illustrates a computer system 1100 upon which an embodiment of the invention may be implemented. Computer system 1100 includes a bus 1102 or other communication mechanism for communicating information, and a processor 1104 coupled with bus 1102 for processing information. Computer system 1100 also includes a main memory 1106, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 1102 for storing information and instructions to be executed by processor 1104. Main memory 1106 also may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 1104. Computer system 1100 further includes a read only memory (ROM) 1108 or other static storage device coupled to bus 1102 for storing static information and instructions for processor 1104. A storage device 1110, such as a magnetic disk or optical disk, is provided and coupled to bus 1102 for storing information and instructions.

Computer system 1100 may be coupled via bus 1102 to a display 1112, such as a cathode ray tube (CRT), for displaying information to a computer user. An input device 1114, including alphanumeric and other keys, is coupled to bus 1102 for communicating information and command selections to processor 1104. Another type of user input device is cursor control 1116, such as a mouse, a trackball, or cursor direction keys for communicating direction information and command selections to processor 1104 and for controlling cursor movement on display 1112. This

input device typically has two degrees of freedom in two axes, a first axis (e.g., x) and a second axis (e.g., y), that allows the device to specify positions in a plane.

The invention is related to the use of computer system 1100 for caching information objects. According to one embodiment of the invention, caching information objects is provided by computer system 1100 in response to processor 1104 executing one or more sequences of one or more instructions contained in main memory 1106. Such instructions may be read into main memory 1106 from another computer-readable medium, such as storage device 1110. Execution of the sequences of instructions contained in main memory 1106 causes processor 1104 to perform the process steps described herein. In alternative embodiments, hard-wired circuitry may be used in place of or in combination with software instructions to implement the invention. Thus, embodiments of the invention are not limited to any specific combination of hardware circuitry and software.

The term "computer-readable medium" as used herein refers to any medium that participates in providing instructions to processor 1104 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 1110. Volatile media includes dynamic memory, such as main memory 1106. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 1102. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications.

Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punch cards, paper tape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read.

Various forms of computer readable media may be involved in carrying one or more sequences of one or more instructions to processor 1104 for execution. For example, the instructions may initially be carried on a magnetic disk of a remote computer. The remote computer can load the instructions into its dynamic memory and send the instructions over a telephone line using a modem. A modem local to computer system 1100 can receive the data on the telephone line and use an infrared transmitter to convert the data to an infrared signal. An infrared detector coupled to bus 1102 can receive the data carried in the infrared signal and place the data on bus 1102. Bus 1102 carries the data to main memory 1106, from which processor 1104 retrieves and executes the instructions. The instructions received by main memory 1106 may optionally be stored on storage device 1110 either before or after execution by processor 1104.

Computer system 1100 also includes a communication interface 1118 coupled to bus 1102. Communication interface 1118 provides a two-way data communication coupling to a network link 1120 that is connected to a local network 1122. For example, communication interface 1118 may be an integrated services digital network (ISDN) card or a modem to provide a data communication connection to a corresponding type of telephone line. As another example, communication interface 1118 may be a local area network (LAN) card to provide a data communication connection to a compatible LAN. Wireless links may also be implemented. In any such implementation, communication interface 1118 sends and receives electrical, electromagnetic or optical signals that carry digital data streams representing various types of information.

Network link 1120 typically provides data communication through one or more networks to other data devices. For example, network link 1120 may provide a connection through local network 1122 to a host computer 1124 or to data equipment operated by an Internet Service Provider (ISP) 1126. ISP 1126 in turn provides data communication services through the world wide packet data communication network now commonly referred to as the "Internet" 1128. Local network 1122 and Internet 1128 both use electrical, electromagnetic or optical signals that carry digital data streams. The signals through the various networks and the signals on network link 1120 and through communication interface 1118, which carry the digital data to and from computer system 1100, are exemplary forms of carrier waves transporting the information.

Computer system 1100 can send messages and receive data, including program code, through the network(s), network link 1120 and communication interface 1118. In the Internet example, a server 1130 might transmit a requested code for an application program through Internet 1128, ISP 1126, local network 1122 and communication interface 1118. In accordance with the invention, one such downloaded application provides for caching information objects as described herein.

The received code may be executed by processor 1104 as it is received, and/or stored in storage device 1110, or other non-volatile storage for later execution. In this manner, computer system 1100 may obtain application code in the form of a carrier wave.

Accordingly, an object cache has been described having distinct advantages over prior approaches. In particular, this document describes an object cache that offers high performance, as measured by low latency and high throughput for object store operations, and large numbers of concurrent operations. The mechanisms described herein are applicable to a large object cache that stores terabytes of information, and billions of objects, commensurate with the growth rate.

The object cache takes advantage of memory storage space efficiency, so expensive semiconductor memory is used sparingly and effectively. The cache also offers disk storage space efficiency, so that large numbers of Internet object replicas can be stored within the finite disk capacity of the object store. The cache is alias free, so that multiple objects or object variants, with different names, but with the same content identical object content, will have the object content cached only once, shared among the different names.

The cache described herein has support for multimedia heterogeneity, efficiently supporting diverse multimedia objects of a multitude of types with size ranging over six orders of magnitude from a few hundred bytes to hundreds of megabytes. The cache has fast, usage-aware garbage collection, so less useful objects can be efficiently removed from the object store to make room for new objects. The cache features data consistency, so programmatic errors and hardware failures do not lead to corrupted data.

The cache has fast restartability, so an object cache can begin servicing requests within seconds of restart, without requiring a time-consuming database or file system check operation. The cache uses streaming I/O, so large objects can be efficiently pipelined from the object store to slow clients, without staging the entire object into memory. The cache has support for content negotiation, so proxy caches can efficiently and flexibly store variants of objects for the same URL, targeted on client browser, language, or other attribute of the client request. The cache is general purpose, so that the object store interface is sufficiently flexible to meet the needs of future media types and protocols.

The foregoing advantages and properties should be regarded as features of the technical description in this document; however, such advantages and properties do not necessarily form a part of the invention, nor are they required by any particular claim that follows this description.

In the foregoing specification, the invention has been described with reference to specific embodiments thereof and with reference to particular goals and advantages. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the invention. The specification and drawings are, accordingly, to be regarded in an illustrative rather than a restrictive sense.

CLAIMS

What is claimed is:

1. In a cache for information objects that are identified by key values based on names of the information objects, comprising a tag table that indexes the information objects using set subkey values based on the key values, a directory table having a plurality of blocks indexed to sets in the tag table by second subkey values based on the key values, and data storage areas referenced by the blocks in the directory table, a method of delivering a requested information object to a client from the cache at a server, comprising the steps of:

(A) receiving a name that identifies a requested information object;

(B) computing a fixed size key value comprising a plurality of subkeys, based on the name;

(C) looking up the requested information object in a directory table, using the subkeys as lookup keys; and

(D) retrieving a copy of the requested information object from the data storage areas using a reference contained in a matching block in the directory table.

2. The method recited in claim 1, further comprising the steps of:

selecting a version of the requested information object from a list in said cache of a plurality of versions of the requested information objects;

identifying a storage location of the requested information object in said cache based on an object key stored in the list in association with the first version;

retrieving the requested information object from the storage location; and

delivering the requested information object to the client.

3. The method recited in claim 1, further comprising the steps of:

storing the information objects contiguously in a mass storage device.

4. The method recited in claim 3, further comprising the steps of:

storing each of the information objects in a contiguous pool of the mass storage device.

5. The method recited in claim 4, further comprising the steps of:

storing each of the information objects in one of a plurality of arenas in the pool.

6. The method recited in claim 5, further comprising the steps of:
storing each of the information objects in one or more fragments, allocated from arenas.

7. The method recited in claim 6, wherein the fragments comprising an information object are linked from the previous fragment key.

8. The method recited in claim 2, further comprising the steps of:
storing the list contiguously with each of the plurality of versions of the requested information object;
in each of the blocks, storing a size value of the requested information object in association with such block, wherein the size value indicates a storage size of the list and the plurality of versions of the information object;
and wherein step (D) comprises the step of reading the list and the plurality of versions concurrently.

9. The method recited in claim 5, further comprising the step of consolidating streaming data transfers of different speeds into a write aggregation buffer.

10. The method recited in claim 3, wherein the step of storing the information objects comprises the step of writing the information objects in contiguous available storage space of the mass storage device, while concurrently performing steps (A) through (D) with respect to another information object.

11. A method of managing a plurality of counters stored in a computer memory, comprising the steps of:
computing a sum of values stored in each of the plurality of counters;
computing a maximum value that can be represented by the plurality of counters; and
decrementing each of the counters when the sum is greater than half the maximum value.

12. The method recited in claim 11, further comprising the steps of:
storing each of the plurality of counters in association with a description of one of a plurality of information objects stored in a cache; and

deleting one of the information objects that is associated with one of the counters when the counter is decremented to zero.

13. The method recited in claim 1, which steps (B) and (C) use subkey partition and table organization, thereby providing fast cache miss determination while using modest memory.

14. A computer-readable medium carrying one or more sequences of instructions for caching information objects, wherein execution of the one or more sequences of instructions by one or more processors causes the one or more processors to perform the steps of:

establishing a cache for information objects that are identified by key values based on names of the information objects, comprising a tag table that indexes the information objects using set subkey values based on the key values, a directory table having a plurality of blocks indexed to sets in the tag table by second subkey values based on the key values, and data storage areas referenced by the blocks in the directory table;

delivering a requested information object to a client from the cache at a server, by:

(A) receiving a name that identifies a requested information object;

(B) computing a fixed size key value comprising a plurality of subkeys, based on the name;

(C) looking up the requested information object in a directory table, using the subkeys as lookup keys; and

(D) retrieving a copy of the requested information object from the data storage areas using a reference contained in a matching block in the directory table.

15. The computer-readable medium recited in claim 14, further comprising the steps of:

selecting a version of the requested information object from a list in said cache of a plurality of versions of the requested information objects;

identifying a storage location of the requested information object in said cache based on an object key stored in the list in association with the first version;

retrieving the requested information object from the storage location; and

delivering the requested information object to the client.

16. The computer-readable medium recited in claim 14, further comprising the steps of:

storing the information objects contiguously in a mass storage device.

17. The computer-readable medium recited in claim 16, further comprising the steps of: storing each of the information objects in a contiguous pool of the mass storage device.

18. The computer-readable medium recited in claim 17, further comprising the steps of: storing each of the information objects in one of a plurality of arenas in the pool.

19. The computer-readable medium recited in claim 18, further comprising the steps of: storing each of the information objects in one or more fragments, allocated from arenas.

20. The computer-readable medium recited in claim 19, wherein the fragments comprising an information object are linked from the previous fragment key.

21. The computer-readable medium recited in claim 15, further comprising the steps of:
storing the list contiguously with each of the plurality of versions of the requested information object;
in each of the blocks, storing a size value of the requested information object in association with such block, wherein the size value indicates a storage size of the list and the plurality of versions of the information object;
and wherein step (D) comprises the step of reading the list and the plurality of versions concurrently.

22. The computer-readable medium recited in claim 18, further comprising the step of consolidating streaming data transfers of different speeds into a write aggregation buffer.

23. The computer-readable medium recited in claim 16, wherein the step of storing the information objects comprises the step of writing the information objects in contiguous available storage space of the mass storage device, while concurrently performing steps (A) through (D) with respect to another information object.

FIGURE 1

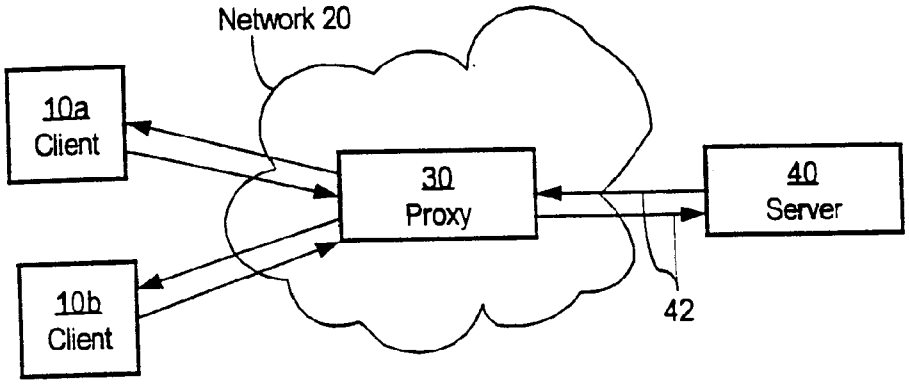


FIGURE 2

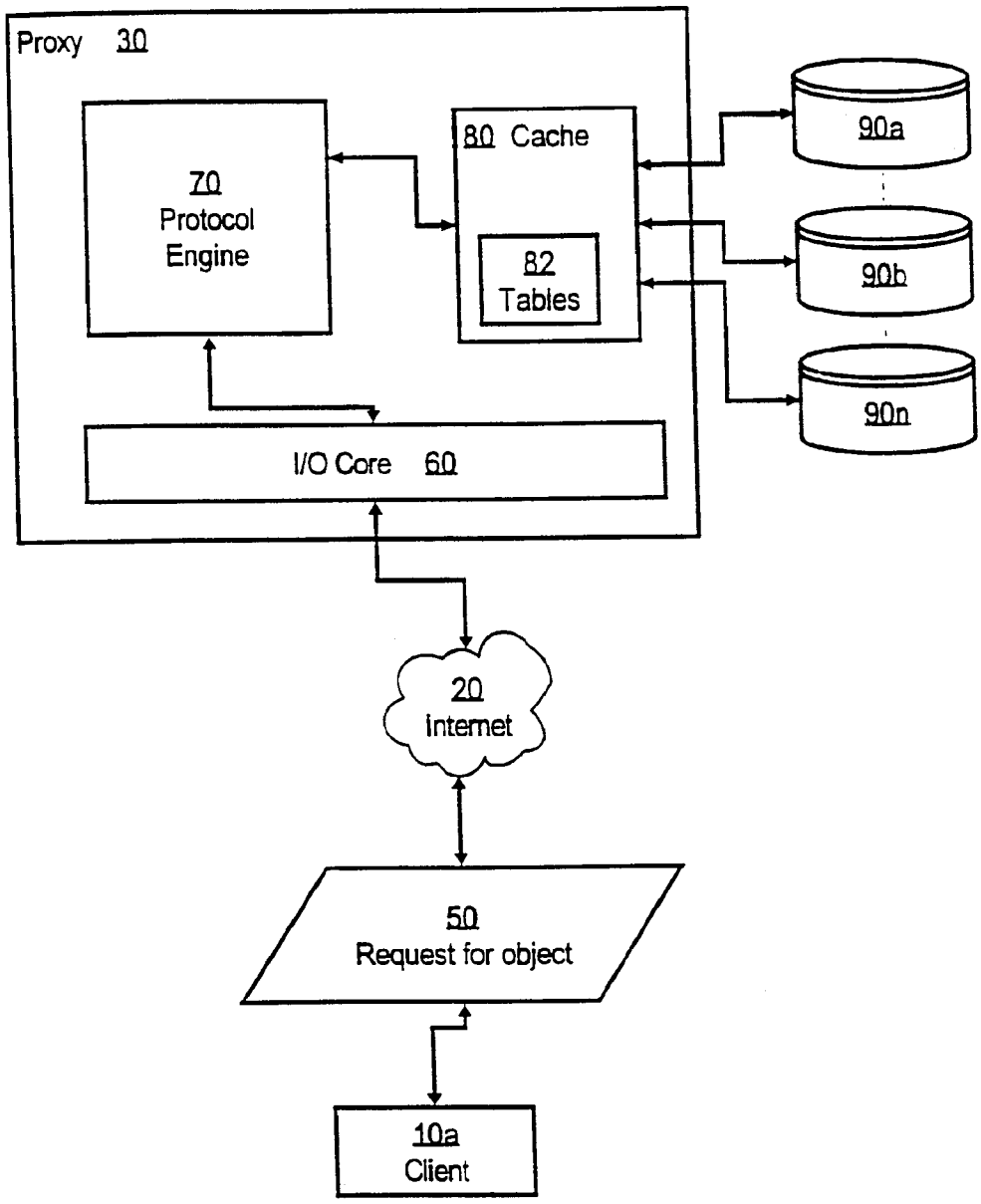

Proxy 30
70
Protocol
Engine
80 Cache
82
Tables
90a
90b
90n
I/O Core 60
20
Internet
50
Request for object
10a
Client

FIGURE 3A

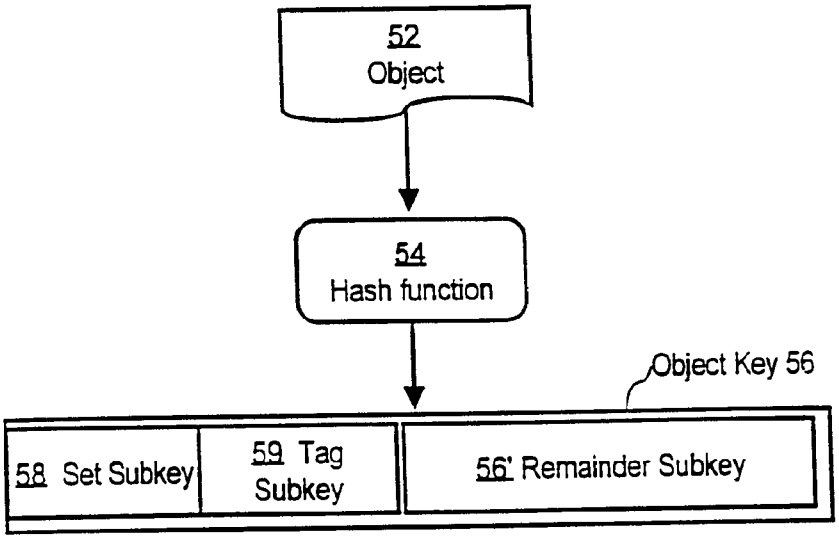
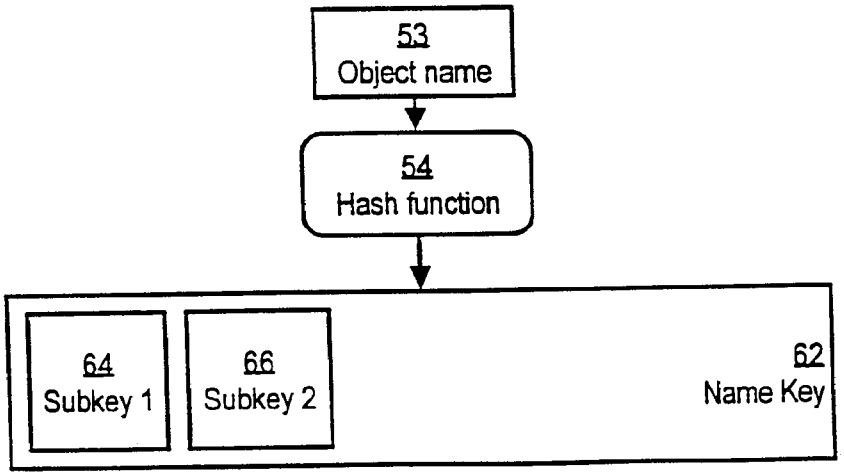


FIGURE 3B



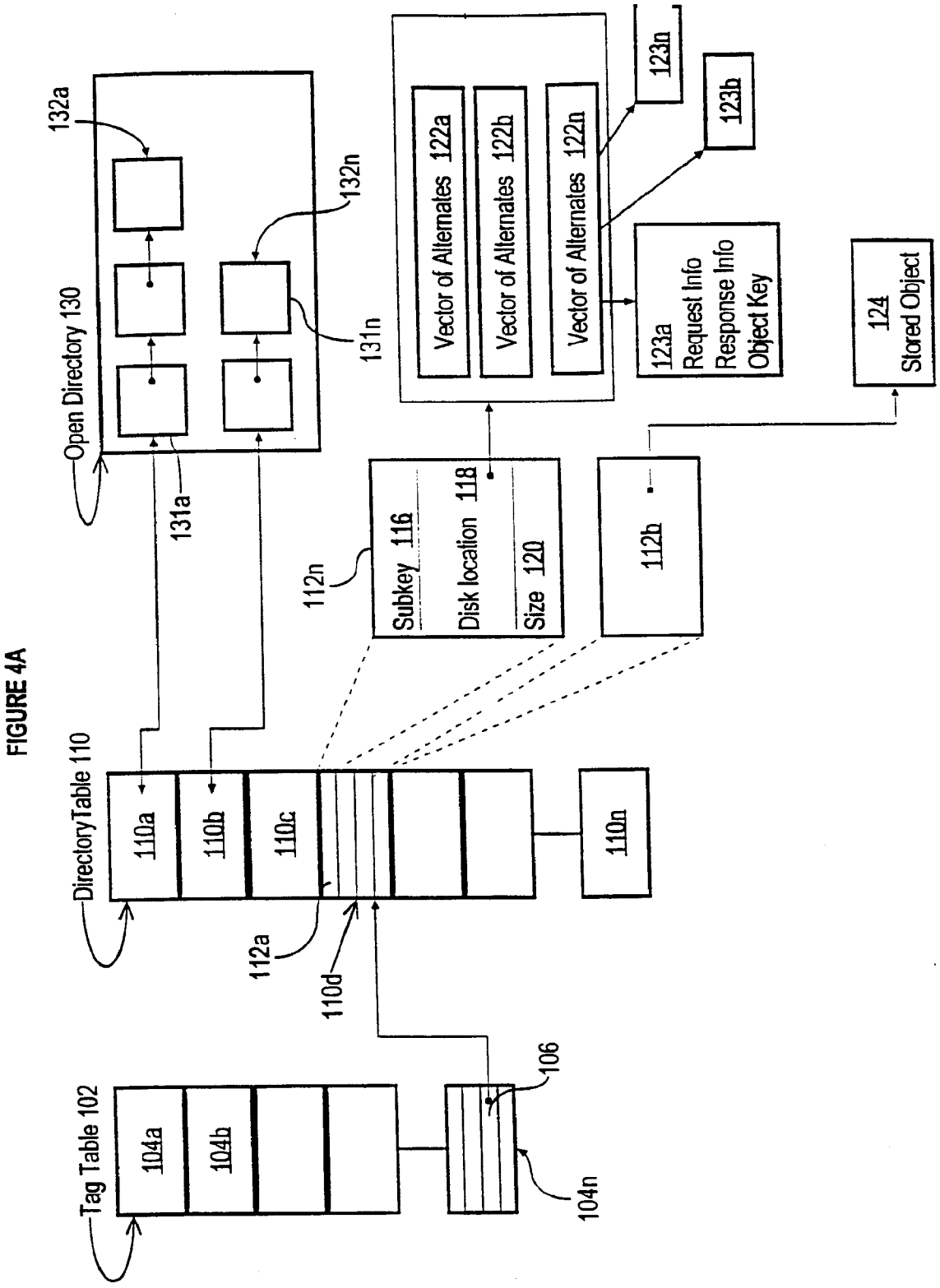

Tag Table 102
104a
104b
104n
106
Directory Table 110
110a
110b
110c
110d
110n
112a
112n
112b
Subkey 116
Disk location 118
Size 120
Open Directory 130
131a
131n
132a
132n
Vector of Alternates 122a
Vector of Alternates 122b
Vector of Alternates 122n
123a
Request Info
Response Info
Object Key
123b
123n
124
Stored Object

FIGURE 4A

FIGURE 4B

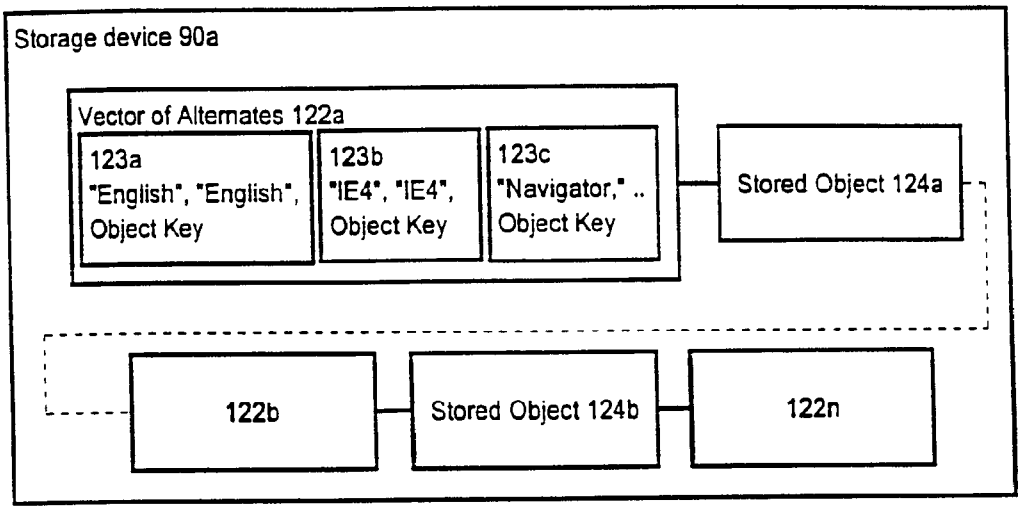

Storage device 90a
Vector of Alternates 122a
123a
"English", "English",
Object Key
123b
"IE4", "IE4",
Object Key
123c
"Navigator," ..
Object Key
Stored Object 124a
122b
Stored Object 124b
122n

FIGURE 5

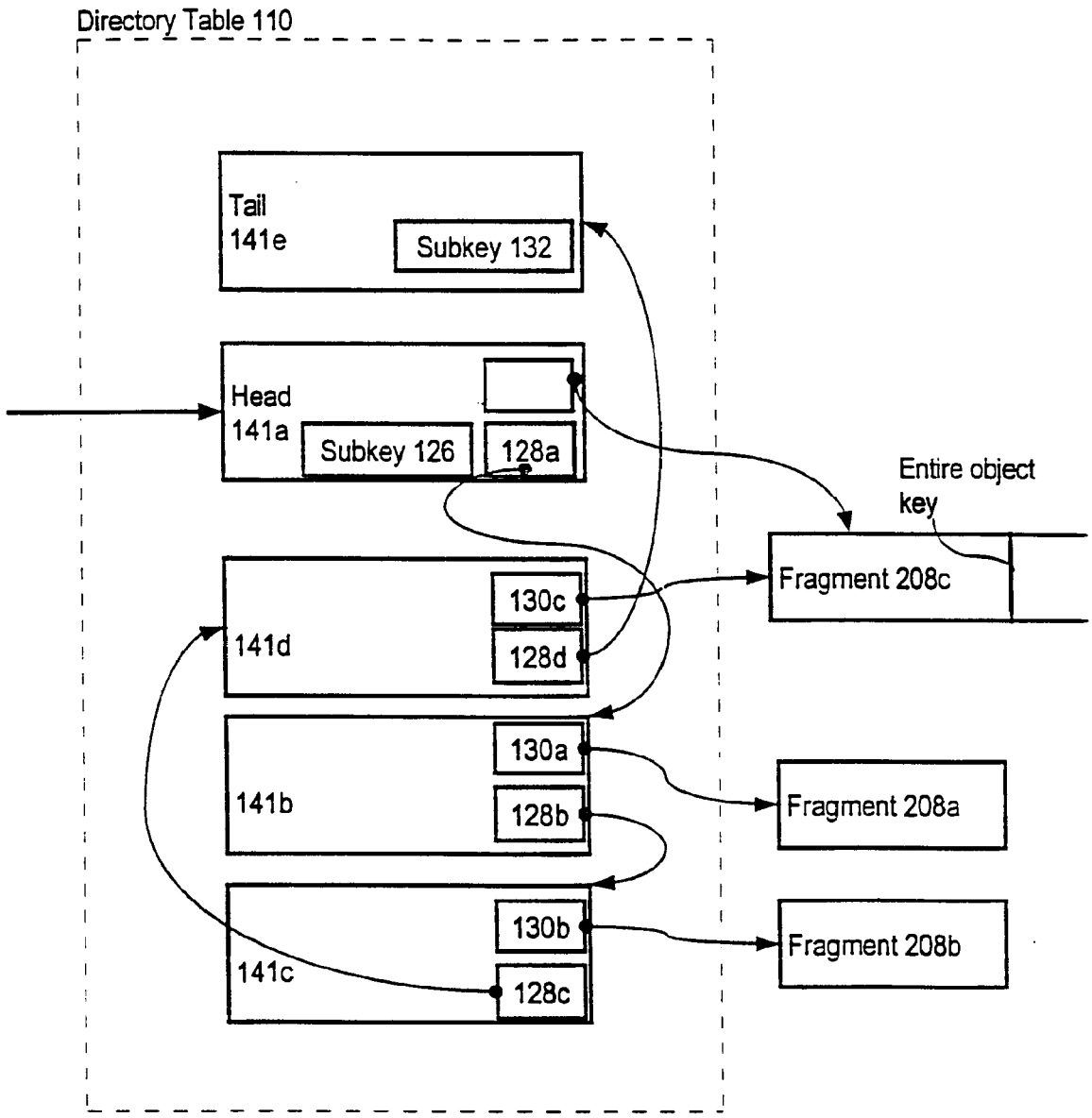

Directory Table 110
Tail
141e
Subkey 132
Head
141a
Subkey 126
128a
Entire object
key
Fragment 208c
130c
141d
128d
130a
141b
128b
Fragment 208a
130b
Fragment 208b
141c
128c

FIGURE 6

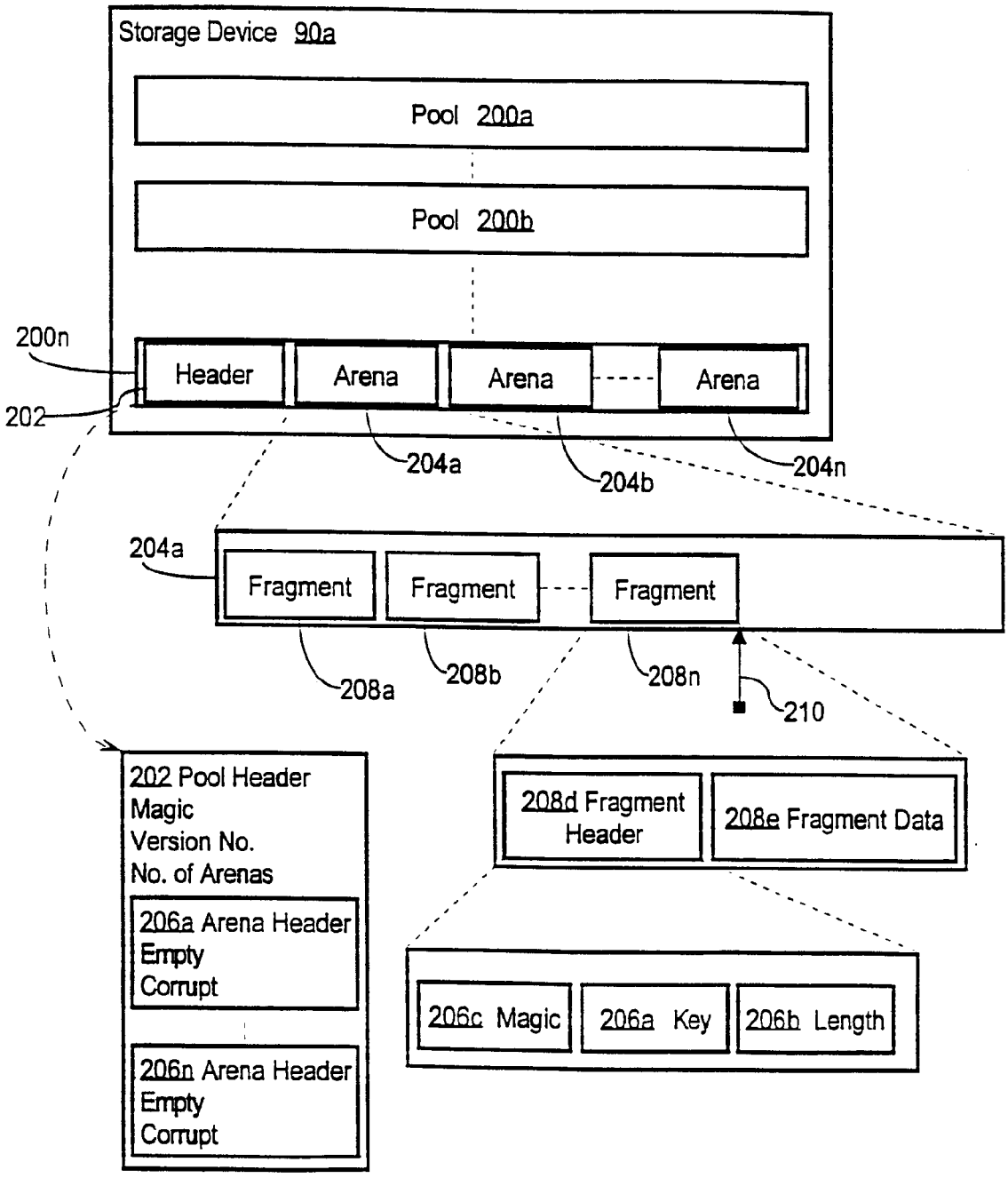

Storage Device 90a
Pool 200a
Pool 200b
200n
Header
Arena
Arena
Arena
202
204a
204b
204n
204a
Fragment
Fragment
Fragment
208a
208b
208n
210
202 Pool Header
Magic
Version No.
No. of Arenas
206a Arena Header
Empty
Corrupt
206n Arena Header
Empty
Corrupt
208d Fragment Header
208e Fragment Data
206c Magic
206a Key
206b Length

FIGURE 7

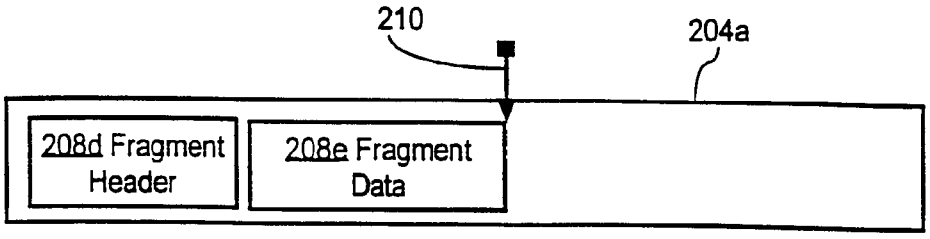


FIGURE 8A

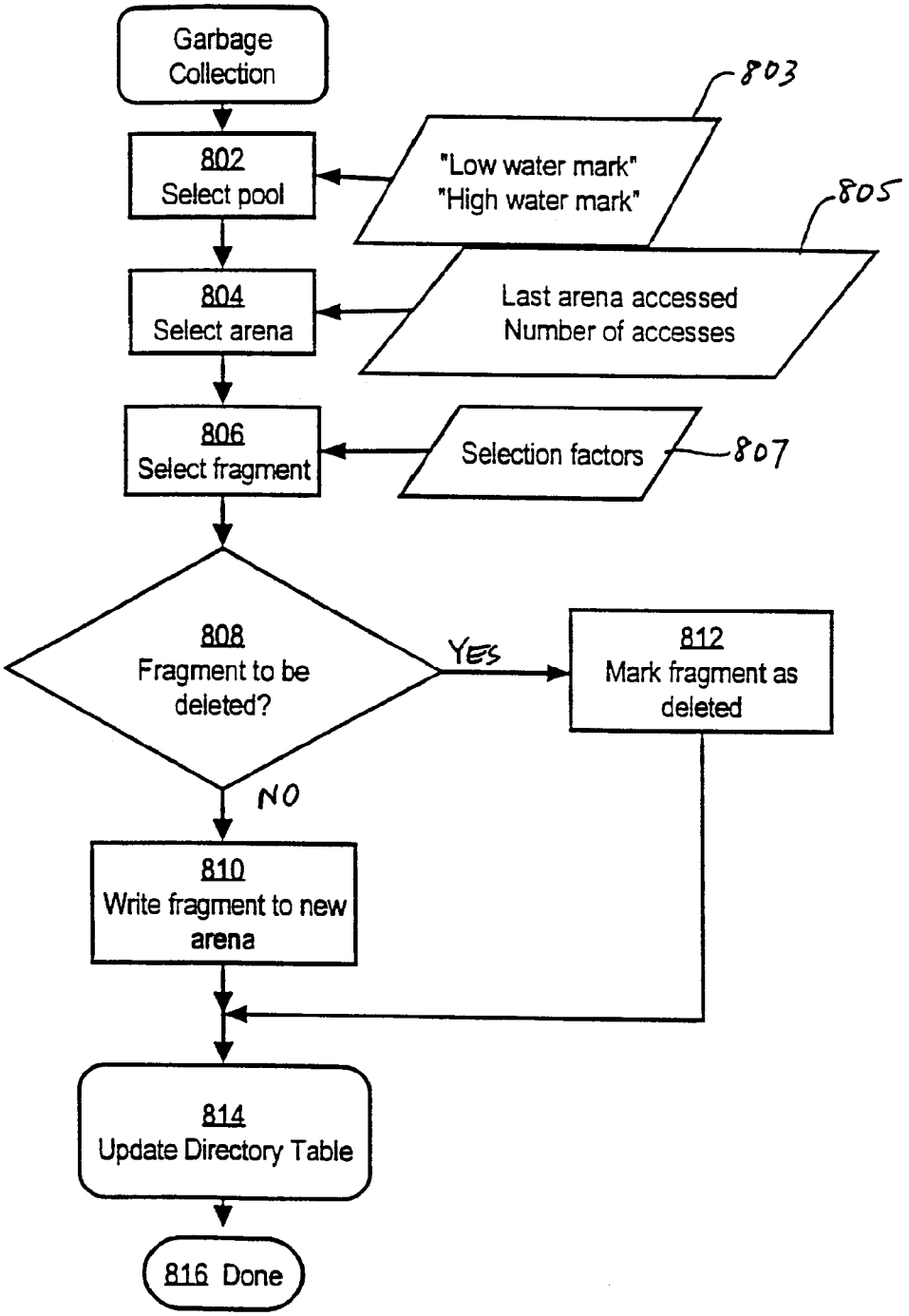

Garbage Collection
802 Select pool
"Low water mark"
"High water mark"
803
805
804 Select arena
Last arena accessed
Number of accesses
806 Select fragment
Selection factors
807
808 Fragment to be deleted?
YES
812 Mark fragment as deleted
NO
810 Write fragment to new arena
814 Update Directory Table
816 Done

FIGURE 8B

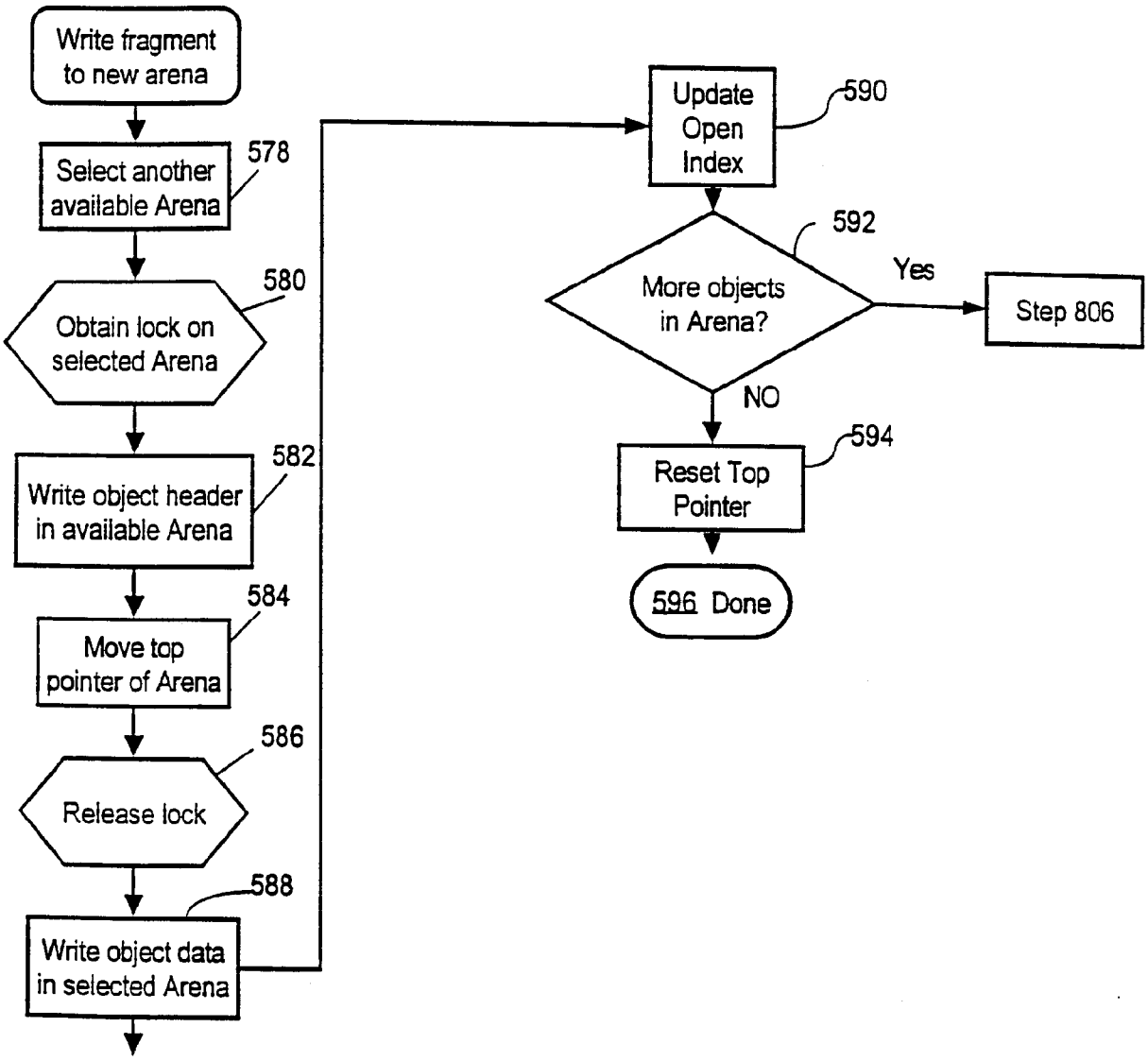

Write fragment to new arena
Select another available Arena
578
Obtain lock on selected Arena
580
Write object header in available Arena
582
Move top pointer of Arena
584
Release lock
586
Write object data in selected Arena
588
Update Open Index
590
More objects in Arena?
592
Yes
Step 806
NO
Reset Top Pointer
594
596 Done

FIGURE 8C

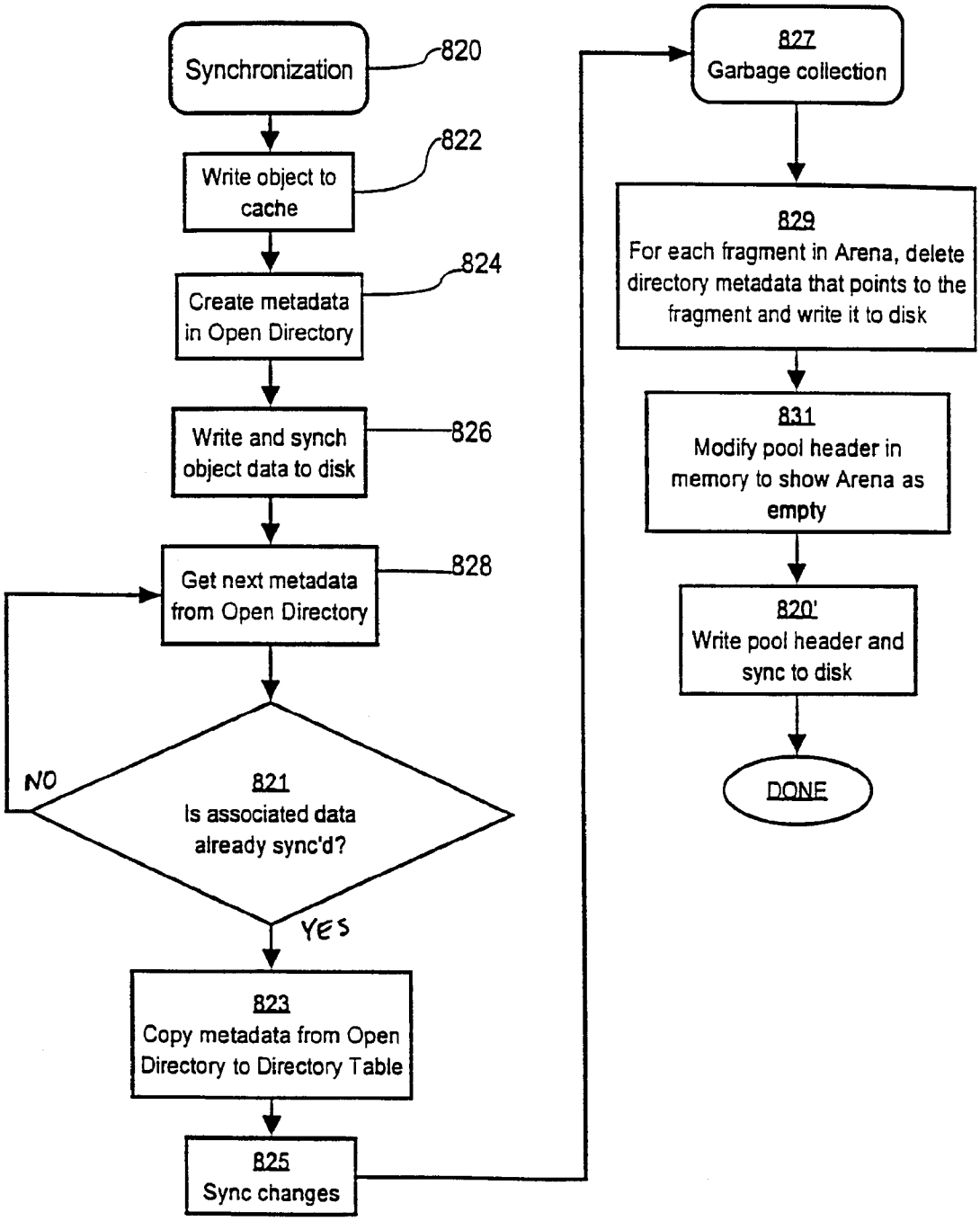

Synchronization
820
Write object to cache
822
Create metadata in Open Directory
824
Write and synch object data to disk
826
Get next metadata from Open Directory
828
821
Is associated data already sync'd?
NO
YES
823
Copy metadata from Open Directory to Directory Table
825
Sync changes
827
Garbage collection
829
For each fragment in Arena, delete directory metadata that points to the fragment and write it to disk
831
Modify pool header in memory to show Arena as empty
820'
Write pool header and sync to disk
DONE

FIGURE 8D

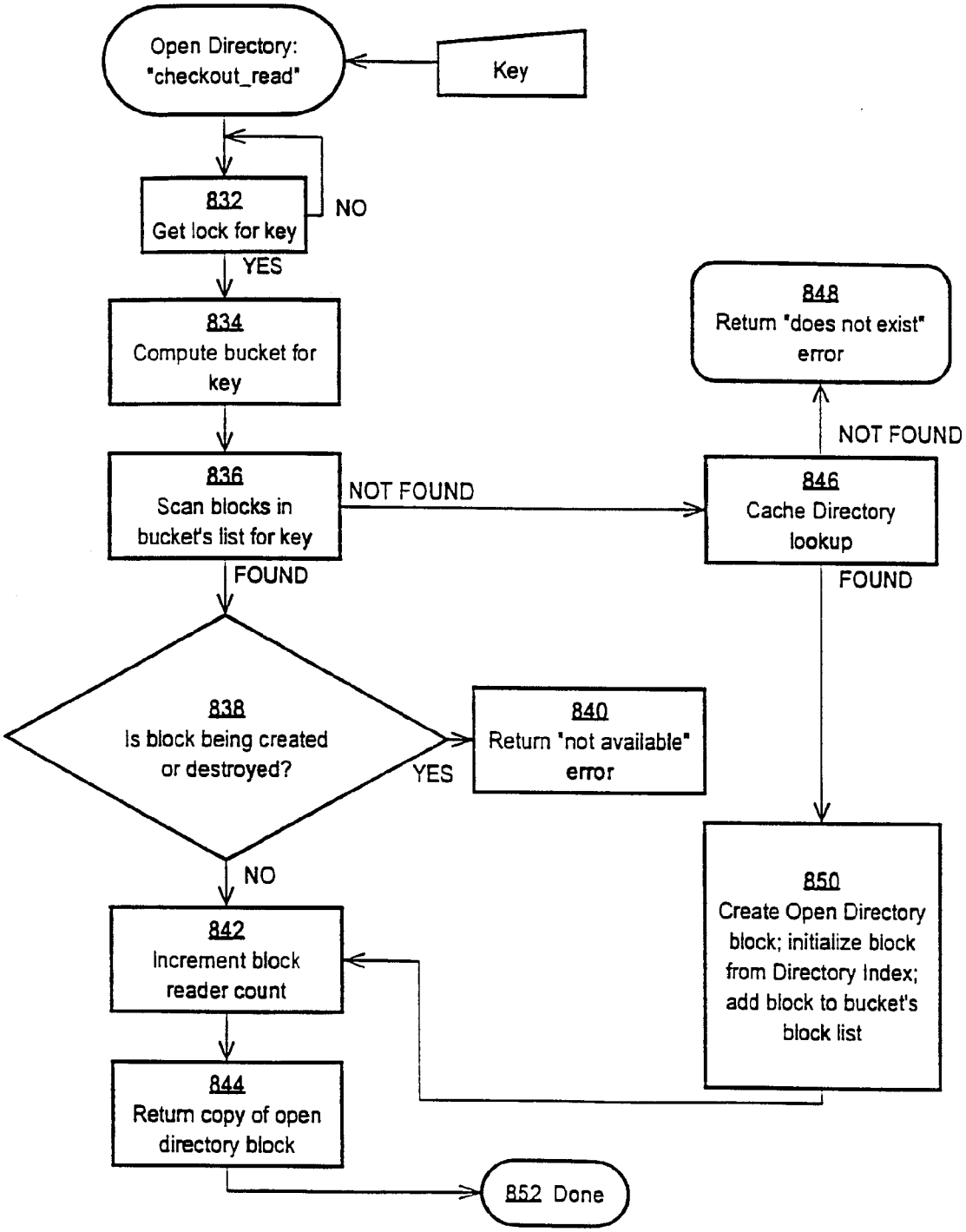

Open Directory: "checkout_read"
Key
832 Get lock for key
NO
YES
834 Compute bucket for key
836 Scan blocks in bucket's list for key
NOT FOUND
FOUND
838 Is block being created or destroyed?
YES
840 Return "not available" error
NO
842 Increment block reader count
844 Return copy of open directory block
852 Done
846 Cache Directory lookup
NOT FOUND
848 Return "does not exist" error
FOUND
850 Create Open Directory block; initialize block from Directory Index; add block to bucket's block list

FIGURE 8E

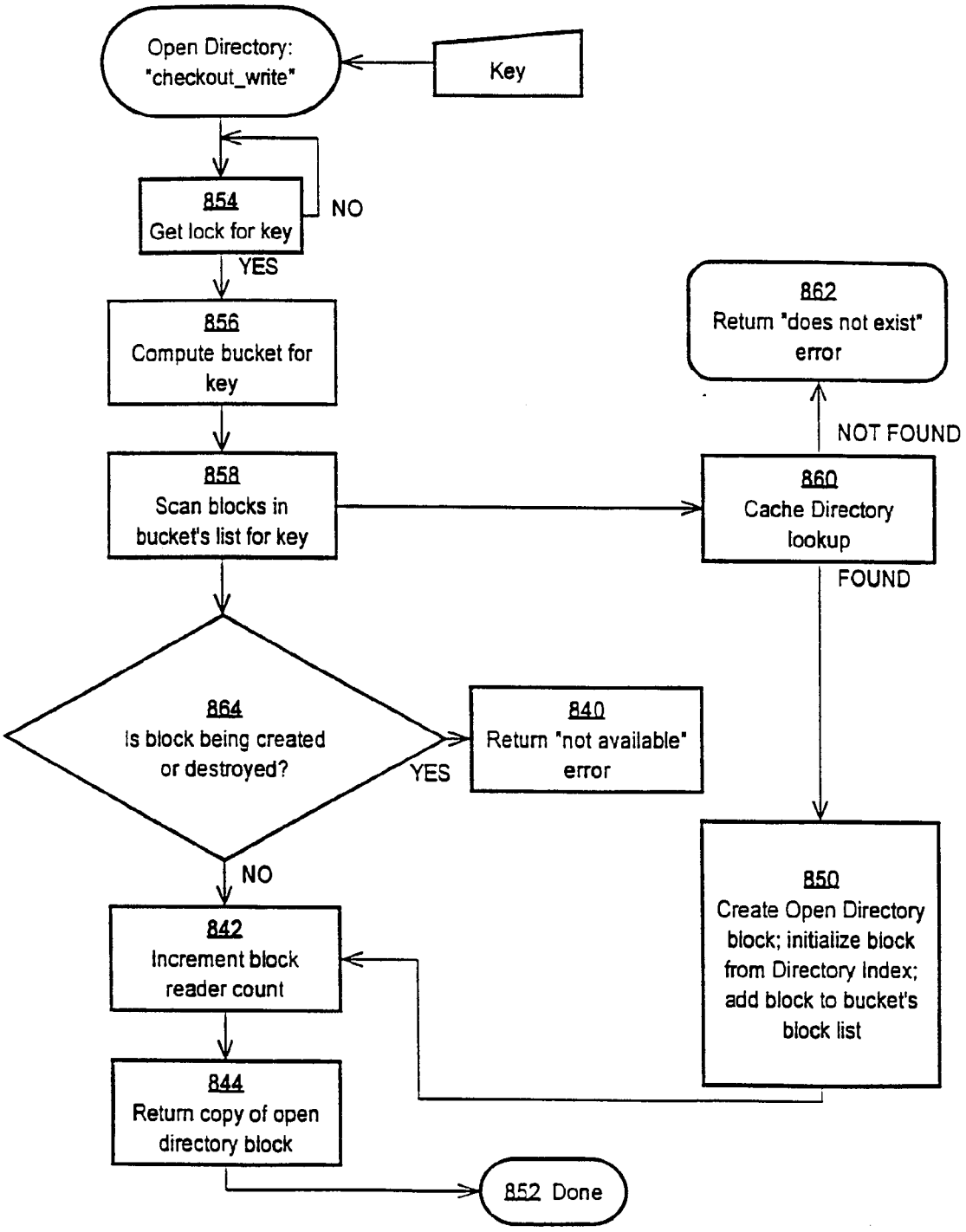

Open Directory: "checkout_write"
Key
854 Get lock for key
NO
YES
856 Compute bucket for key
858 Scan blocks in bucket's list for key
862 Return "does not exist" error
NOT FOUND
860 Cache Directory lookup
FOUND
864 Is block being created or destroyed?
YES
840 Return "not available" error
NO
842 Increment block reader count
850 Create Open Directory block; initialize block from Directory Index; add block to bucket's block list
844 Return copy of open directory block
852 Done

FIGURE 8F

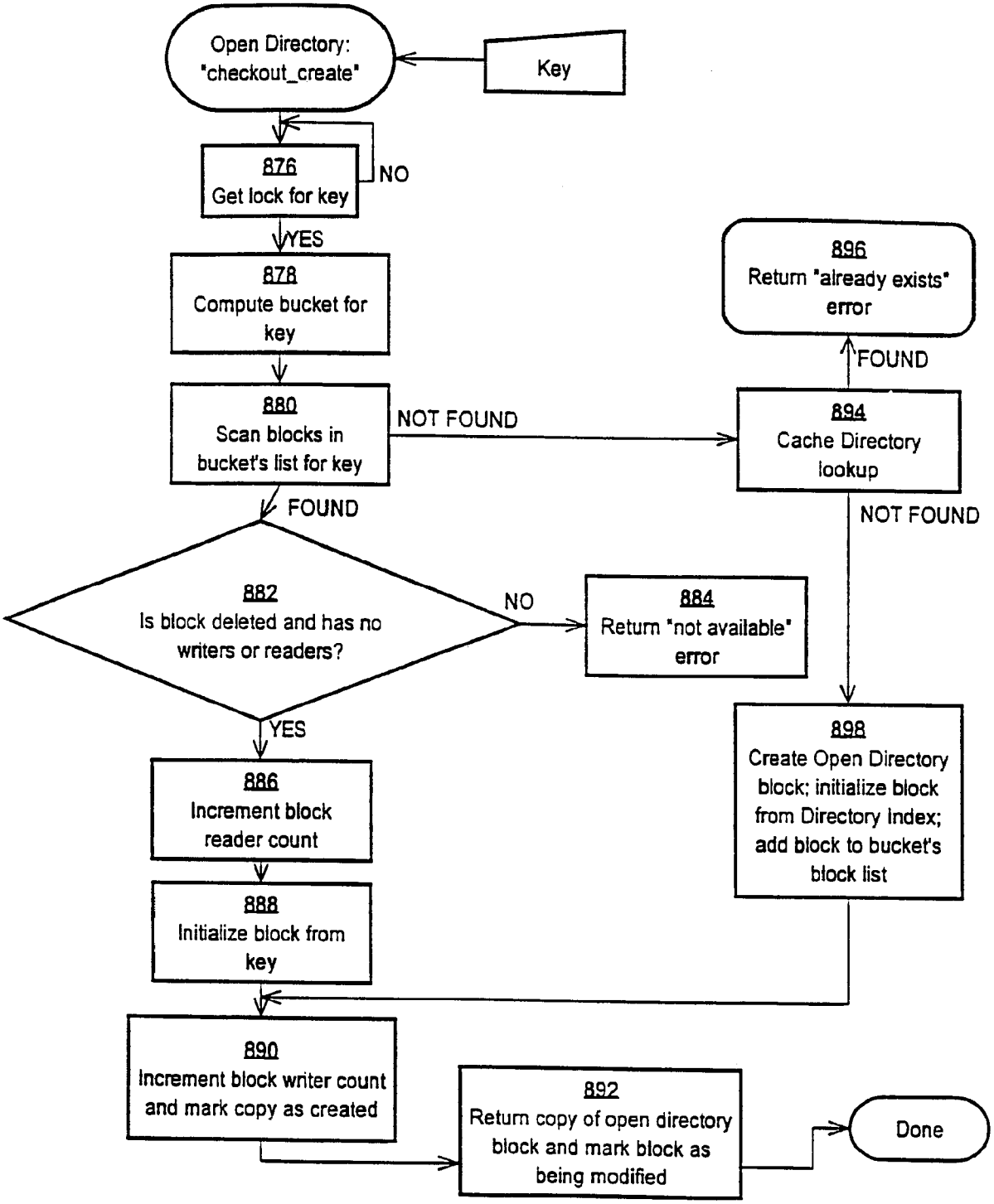

Open Directory: "checkout_create"
Key
876 Get lock for key
NO
YES
878 Compute bucket for key
880 Scan blocks in bucket's list for key
NOT FOUND
FOUND
882 Is block deleted and has no writers or readers?
NO
884 Return "not available" error
YES
886 Increment block reader count
888 Initialize block from key
890 Increment block writer count and mark copy as created
892 Return copy of open directory block and mark block as being modified
Done
894 Cache Directory lookup
FOUND
896 Return "already exists" error
NOT FOUND
898 Create Open Directory block; initialize block from Directory Index; add block to bucket's block list

FIGURE 9A

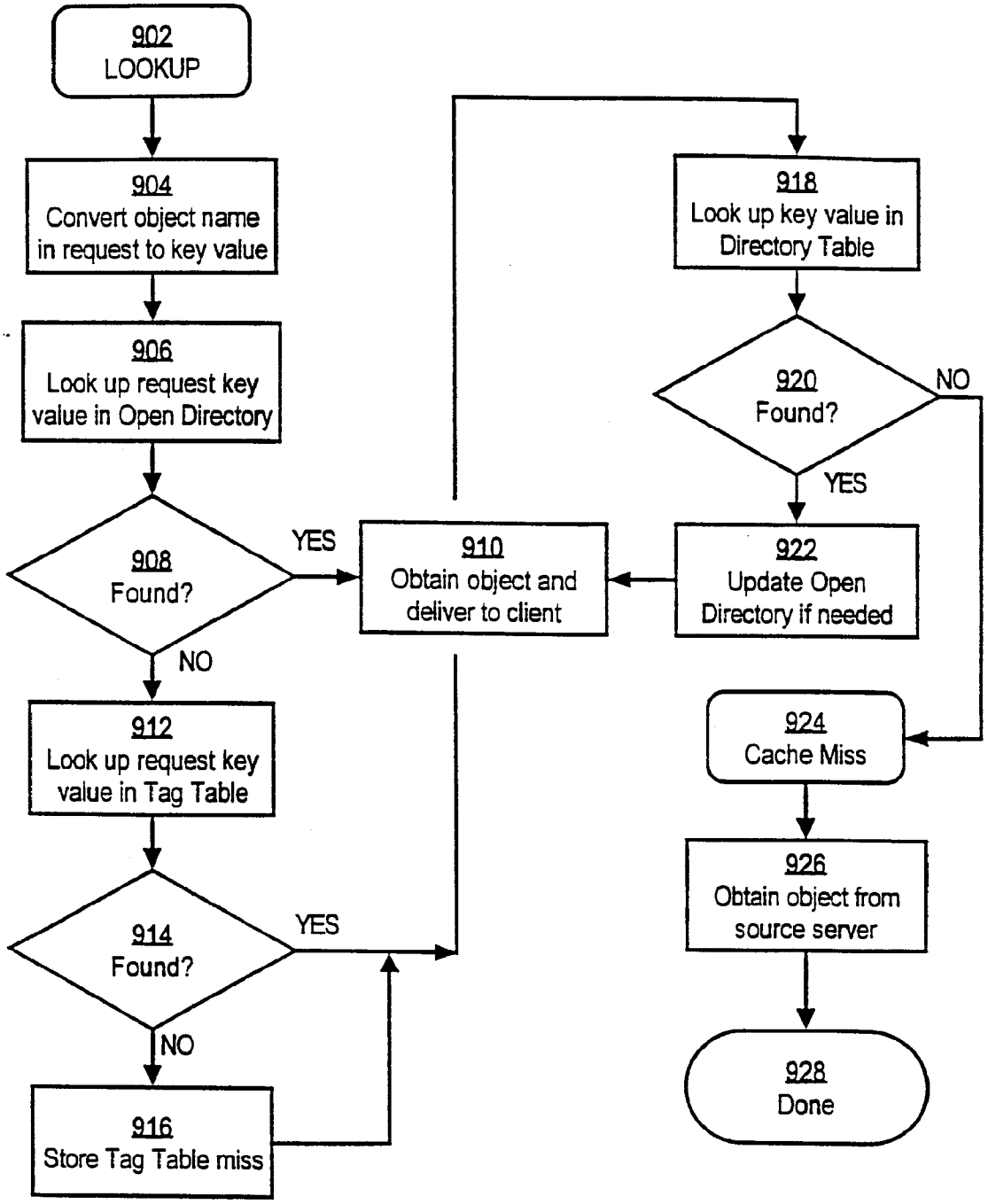

902
LOOKUP
904
Convert object name in request to key value
906
Look up request key value in Open Directory
908
Found?
YES
NO
910
Obtain object and deliver to client
912
Look up request key value in Tag Table
914
Found?
YES
NO
916
Store Tag Table miss
918
Look up key value in Directory Table
920
Found?
NO
YES
922
Update Open Directory if needed
924
Cache Miss
926
Obtain object from source server
928
Done

FIGURE 9B

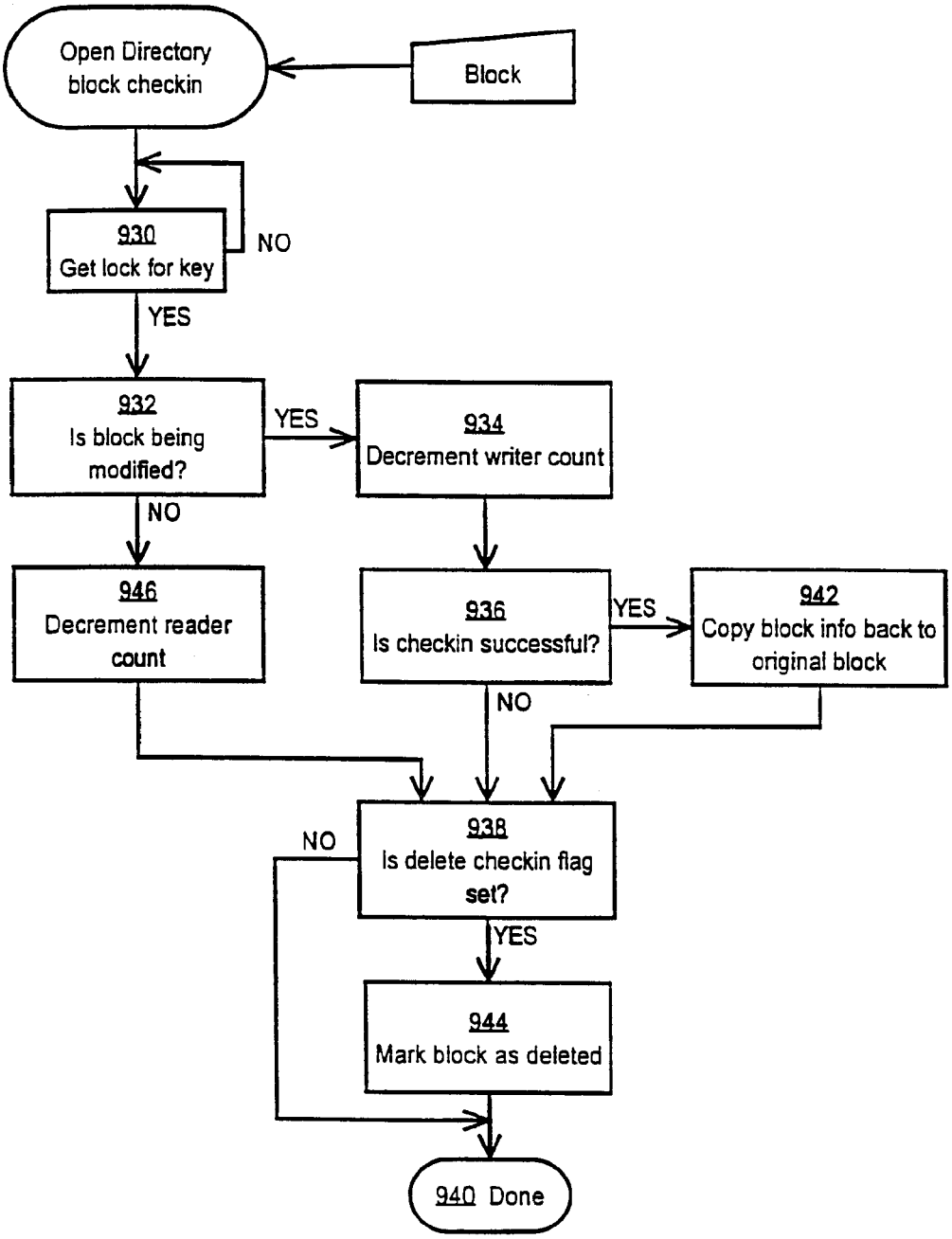

Open Directory block checkin
Block
930 Get lock for key
NO
YES
932 Is block being modified?
YES
934 Decrement writer count
NO
946 Decrement reader count
936 Is checkin successful?
YES
942 Copy block info back to original block
NO
938 Is delete checkin flag set?
NO
YES
944 Mark block as deleted
940 Done

FIGURE 9C

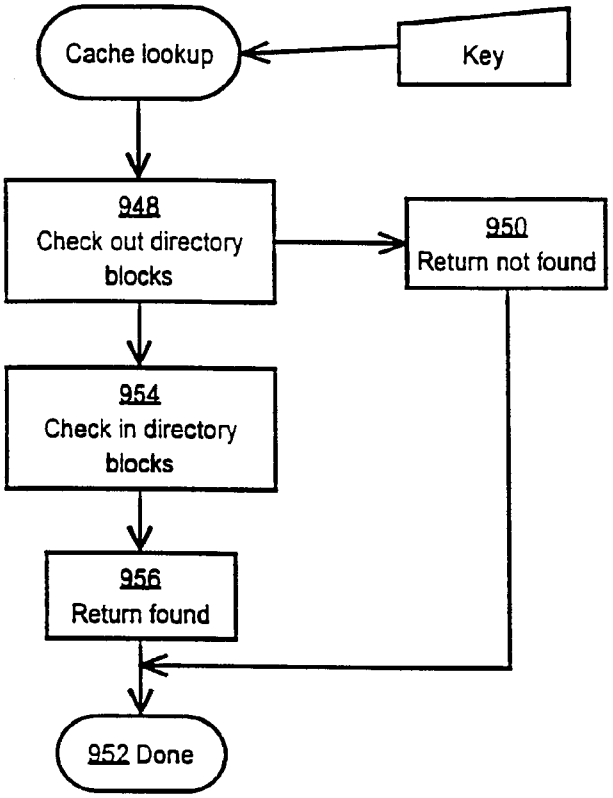

Cache lookup
Key
948
Check out directory blocks
950
Return not found
954
Check in directory blocks
956
Return found
952 Done

FIGURE 9D

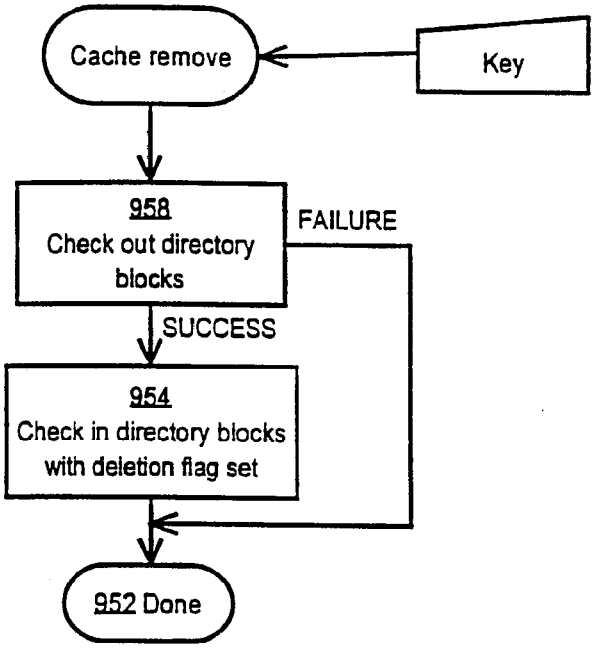

Cache remove
Key
958
Check out directory blocks
FAILURE
SUCCESS
954
Check in directory blocks with deletion flag set
952 Done

FIGURE 9E

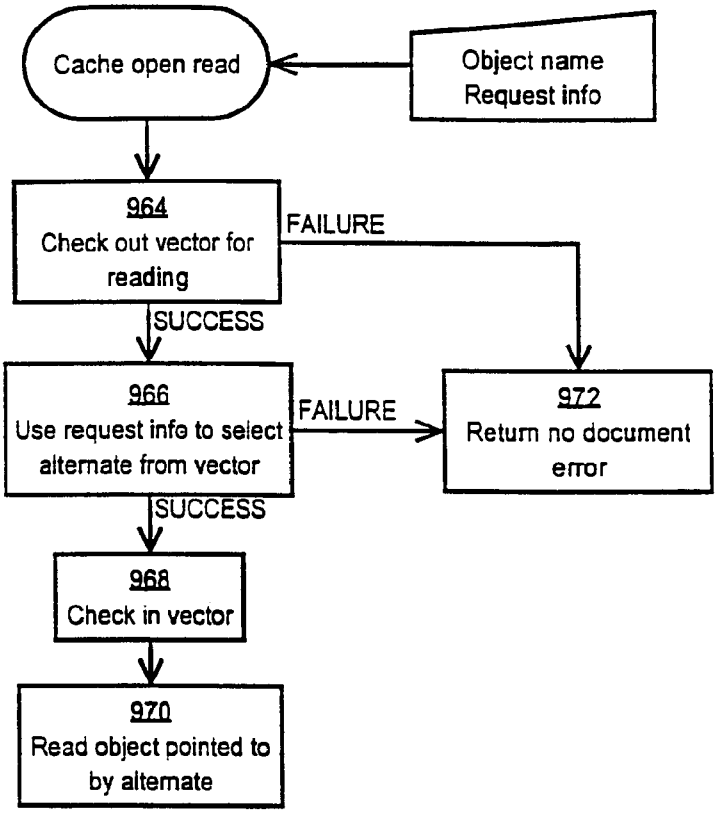

Cache open read
Object name
Request info
964
Check out vector for reading
FAILURE
SUCCESS
966
Use request info to select alternate from vector
FAILURE
972
Return no document error
SUCCESS
968
Check in vector
970
Read object pointed to by alternate

FIGURE 9F

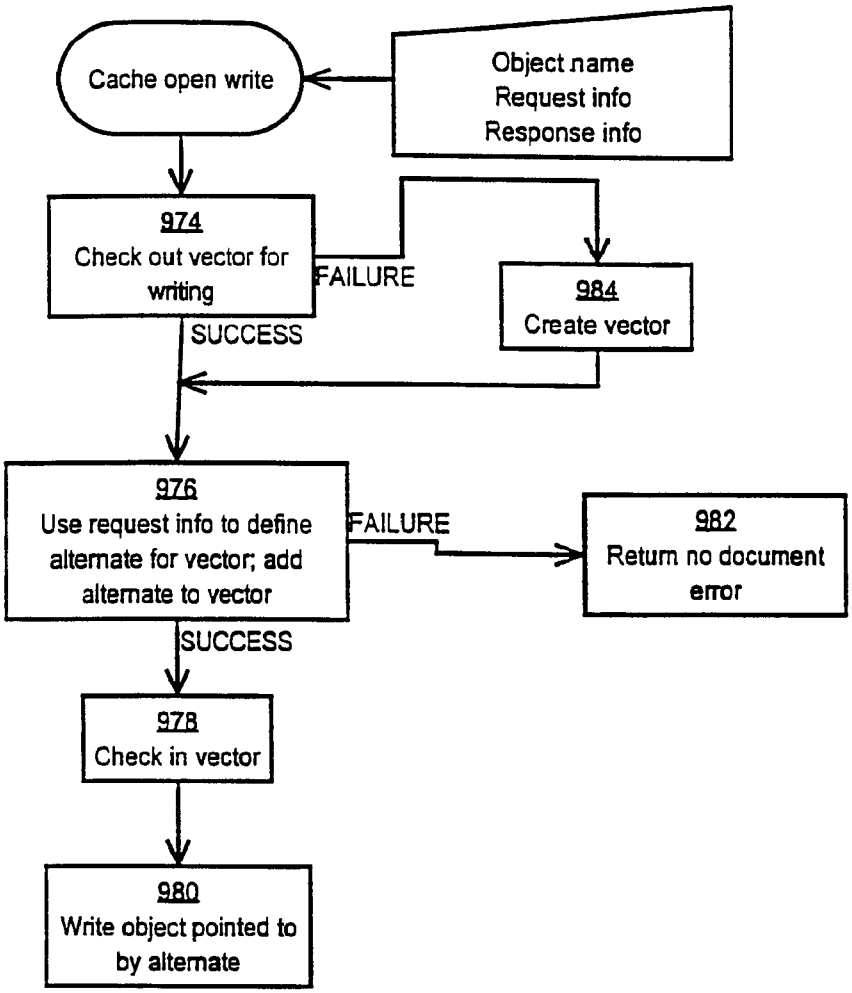

Cache open write
Object name
Request info
Response info
974
Check out vector for writing
FAILURE
984
Create vector
SUCCESS
976
Use request info to define alternate for vector; add alternate to vector
FAILURE
982
Return no document error
SUCCESS
978
Check in vector
980
Write object pointed to by alternate

FIGURE 9G

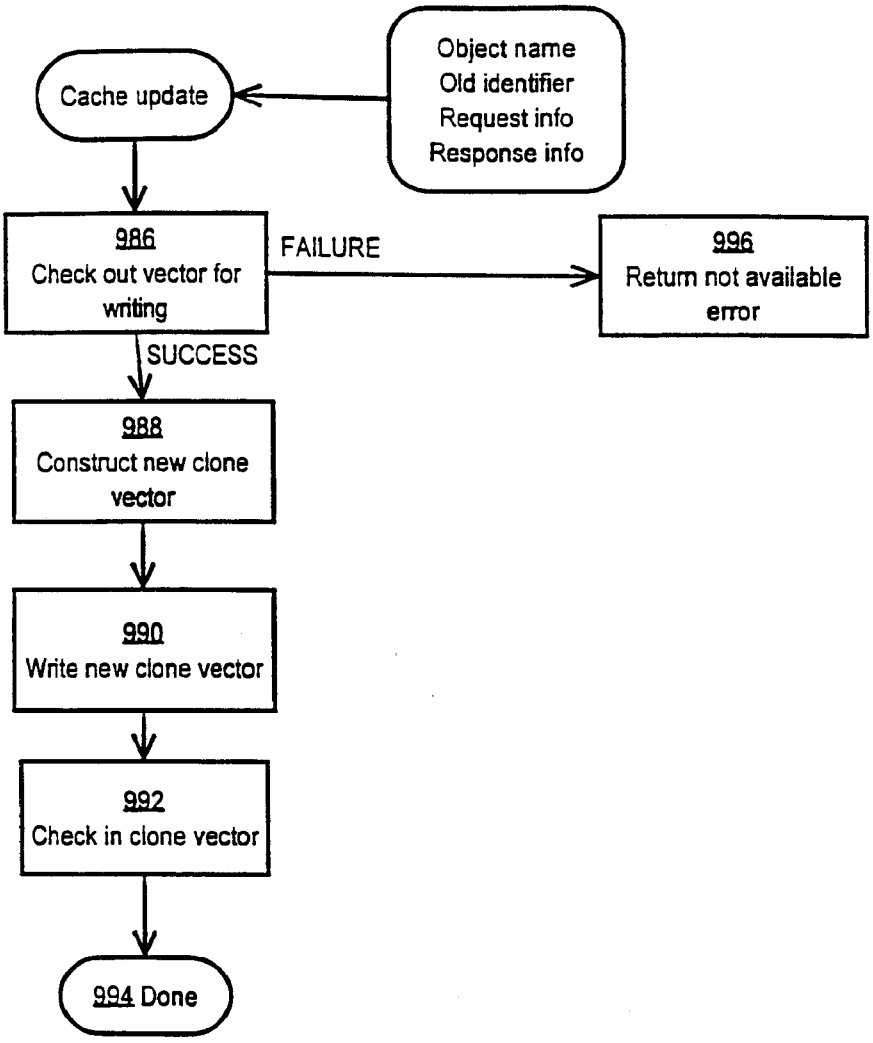

Cache update
Object name
Old identifier
Request info
Response info
986
Check out vector for writing
FAILURE
996
Return not available error
SUCCESS
988
Construct new clone vector
990
Write new clone vector
992
Check in clone vector
994 Done

FIGURE 10A

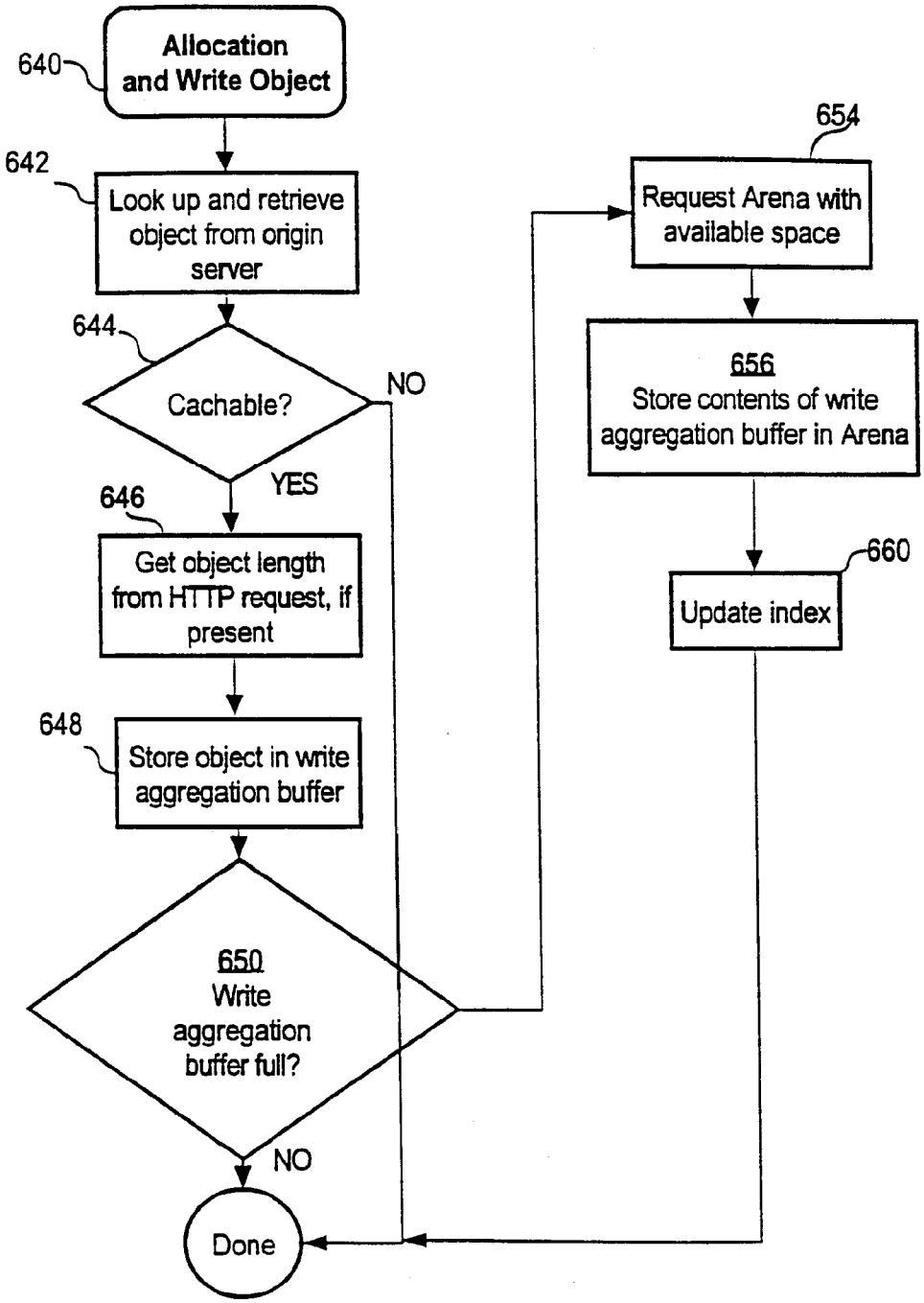
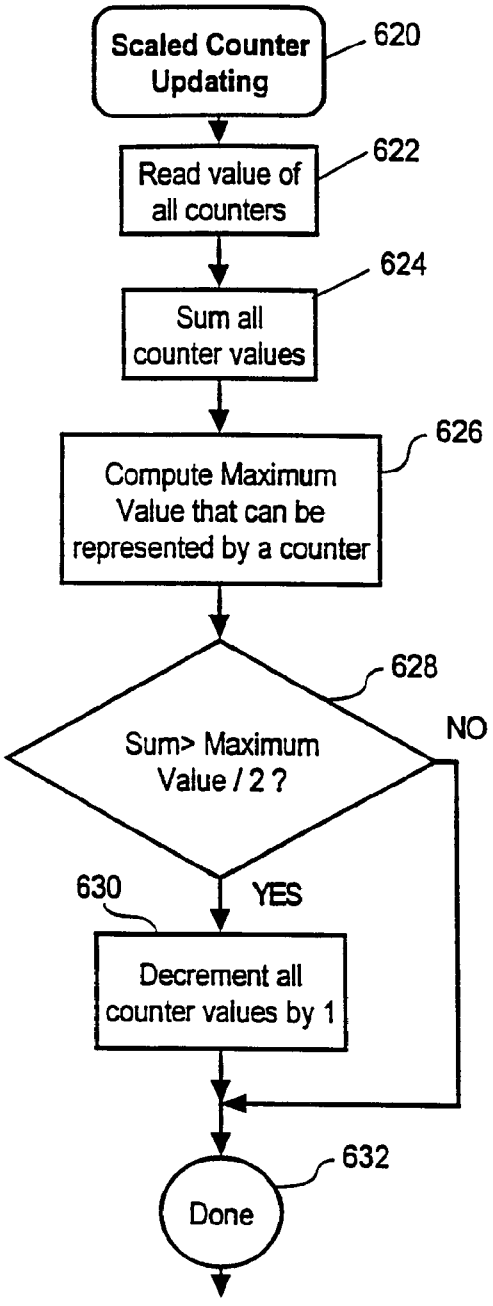

640
Allocation and Write Object
642
Look up and retrieve object from origin server
644
Cachable?
NO
YES
646
Get object length from HTTP request, if present
648
Store object in write aggregation buffer
650
Write aggregation buffer full?
NO
Done
654
Request Arena with available space
656
Store contents of write aggregation buffer in Arena
660
Update index

FIGURE 10B


Scaled Counter Updating
620
Read value of all counters
622
Sum all counter values
624
Compute Maximum Value that can be represented by a counter
626
Sum> Maximum Value / 2 ?
628
NO
YES
630
Decrement all counter values by 1
Done
632

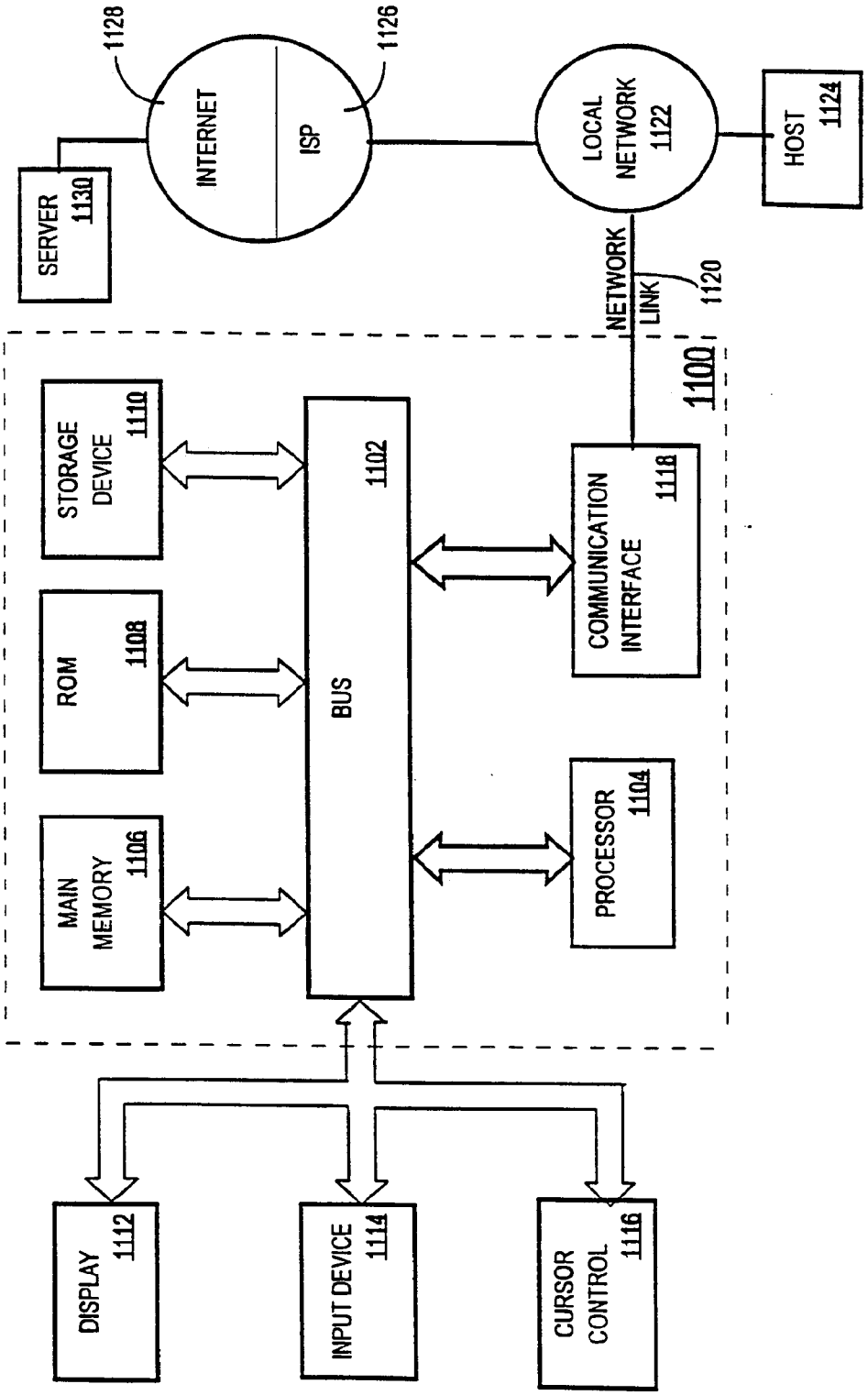
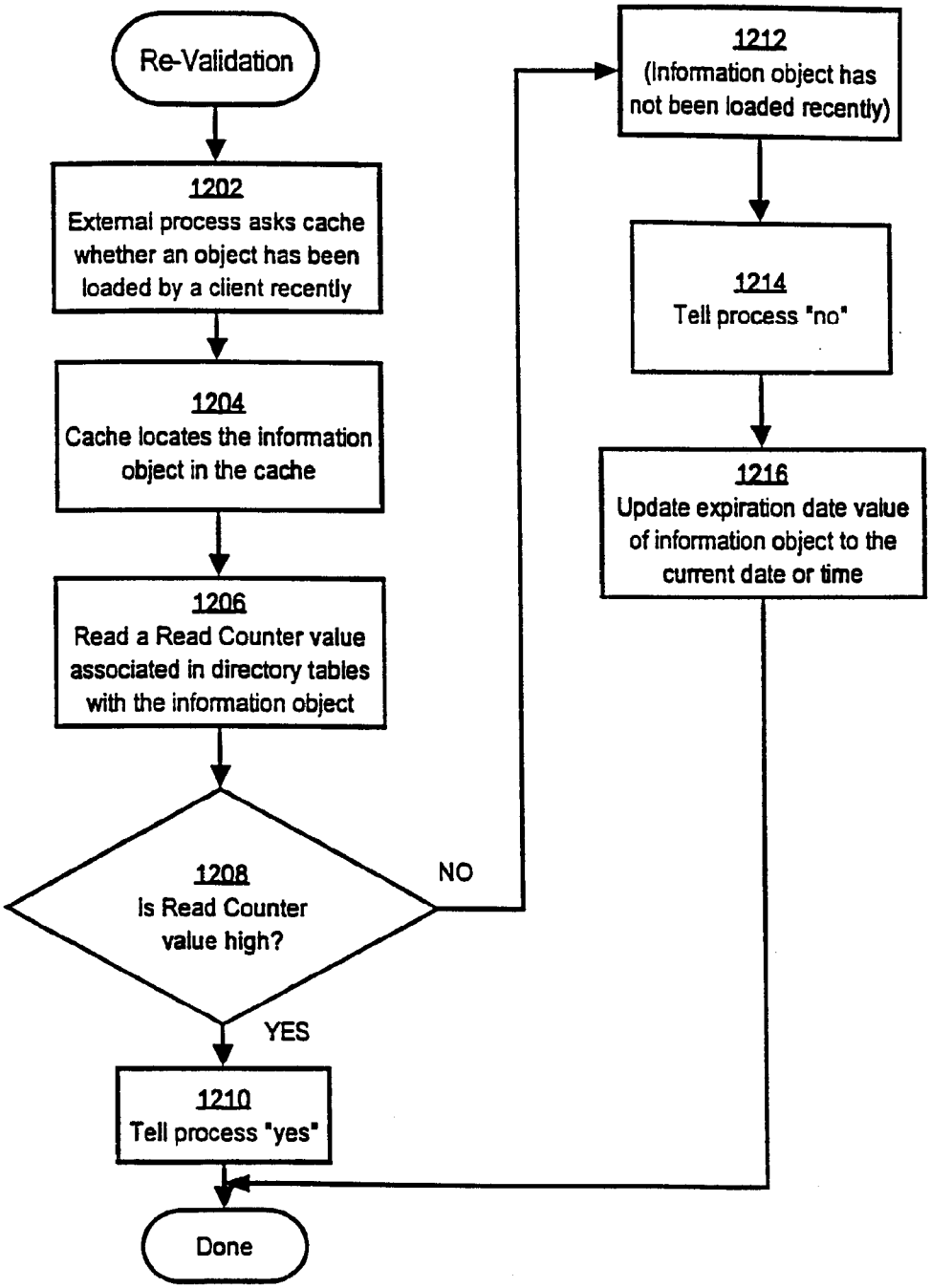

1128
1126
INTERNET
ISP
LOCAL NETWORK 1122
HOST 1124
SERVER 1130
NETWORK LINK 1120
1100
STORAGE DEVICE 1110
ROM 1108
MAIN MEMORY 1106
BUS 1102
COMMUNICATION INTERFACE 1118
PROCESSOR 1104
DISPLAY 1112
INPUT DEVICE 1114
CURSOR CONTROL 1116

FIG. 11

FIGURE 12


Re-Validation
1202
External process asks cache whether an object has been loaded by a client recently
1204
Cache locates the information object in the cache
1206
Read a Read Counter value associated in directory tables with the information object
1208
Is Read Counter value high?
NO
YES
1210
Tell process "yes"
Done
1212
(Information object has not been loaded recently)
1214
Tell process "no"
1216
Update expiration date value of information object to the current date or time

INTERNATIONAL SEARCH REPORT

Intern ıal Application No
PCT/US 99/08281

A. CLASSIFICATION OF SUBJECT MATTER
IPC 6 G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 6 G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	EP 0 834 819 A (SUN MICROSYSTEMS INC) 8 April 1998 (1998-04-08) abstract page 2, line 1 - page 2, line 57	1-23
A	US 5 732 267 A (SMITH MATTHEW W) 24 March 1998 (1998-03-24) abstract column 1, line 1 - column 3, line 32	1-23
A	EP 0 811 939 A (WEBTV NETWORKS INC) 10 December 1997 (1997-12-10) abstract column 1, line 1 - column 2, line 49	1-23

☐ Further documents are listed in the continuation of box C.

☒ Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
17 August 1999	23/08/1999

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Katerbau, R

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

Internal Application No
PCT/US 99/08281

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0834819 A	08-04-1998	JP 10232814 A	02-09-1998
US 5732267 A	24-03-1998	NONE	
EP 0811939 A	10-12-1997	US 5918013 A	29-06-1999
		AU 3375197 A	05-01-1998
		JP 10228437 A	25-08-1998
		WO 9746943 A	11-12-1997

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30, H04L 29/08 — A1

(11) International Publication Number: WO 00/29990

(43) International Publication Date: 25 May 2000 (25.05.00)

(21) International Application Number: PCT/US99/27325

(22) International Filing Date: 17 November 1999 (17.11.99)

(30) Priority Data:
60/108,855 18 November 1998 (18.11.98) US
09/440,975 16 November 1999 (16.11.99) US

(71) Applicant: INFOLIBRIA, INC. [US/US]; Suite 323, 411 Waverly Oaks Road, Waltham, MA 02454 (US).

(72) Inventors: LEWIS, Kevin, T.; 100 Fulton Street, No 4T, Boston, MA 02109 (US). HOEY, David, L.; 102 Concord Road, Wayland, MA 01778 (US).

(74) Agents: THIBODEAU, David, J., Jr. et al.; Hamilton, Brook, Smith & Reynolds, P.C., Two Militia Drive, Lexington, MA 02421 (US).

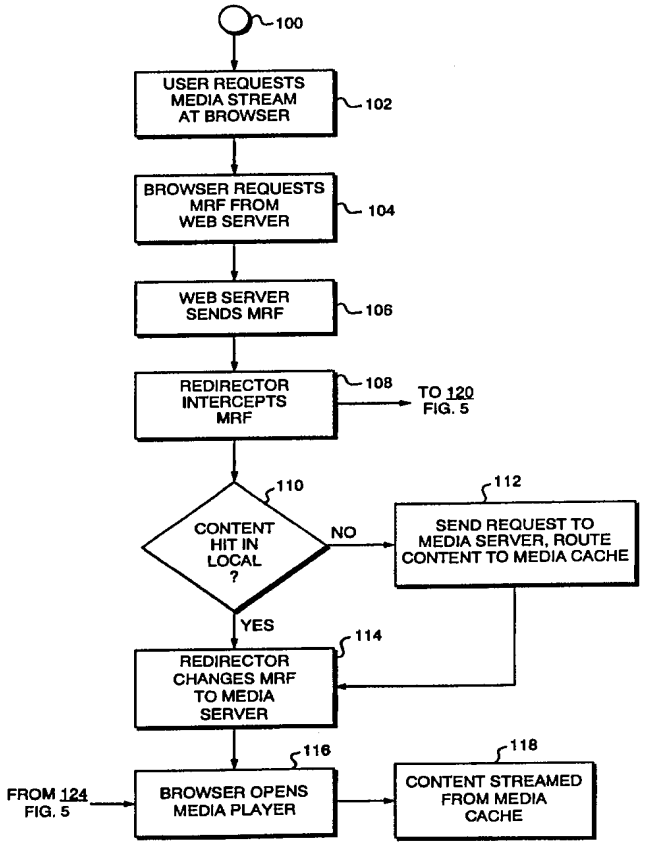
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: EFFICIENT CONTENT SERVER USING REQUEST REDIRECTION

(57) Abstract

A technique for optimizing delivery of specialized content files such as multimedia files in a computer network. A multimedia file cache is located at a point of presence, such as at an Internet Service Provider (ISP), gateway, or other place close to user connection points into the network. A redirection function intercepts media redirection file requests sent to or from the media server and rewrites such requests so that they point to the local media file cache.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

EFFICIENT CONTENT SERVER USING REQUEST REDIRECTION

FIELD OF THE INVENTION

This invention relates generally to content delivery in computer networks and in particular to a technique for selectively redirecting media content requests to local caches.

BACKGROUND

During the twentieth century, continuous media services such as broadcast and cable television have become an ubiquitous vehicle for distributing entertainment, education, advertising, and other programs to the public. Emerging standards define new formats for television broadcast distribution in digital form. In addition, devices such as cable modems and Digital Subscriber Line (xDSL) modems are beginning to become available to the public. These products promise to provide a mix of access to television programming and high speed telecommunication services such as the Internet, which are increasingly looked to as vehicles which will augment traditional television broadcasts.

Indeed, these two content delivery systems, those optimized for television broadcast and those optimized for distribution of data content are merging. For example, standards such as the Motion Picture Experts Group (MPEG)-2 standard and other encoding schemes specify how video signals may be digitized for efficient transfer. Other efforts such as the Internet Engineering Task Force (IETF) audio/video transport (AVT) working group are developing specifications for realtime transmission of video over User Datagram

Protocol (UDP) and Internet Protocol (IP) type layered networks such as the Internet. The vision is that in the future it will not be necessary to have an analog television within the home or at another cable subscriber site. In particular, video programs will routinely be broadcast in digital form as packetized data using UDP/IP or other data protocols.

In addition, with the growth of the Web and other activities on the Internet, it is increasingly important to find ways to efficiently manage a mix of text, graphics, and video data such as is associated with typical Web browsing activities. New techniques for specifying, handling, and distributing multimedia content such as contained in video clips are therefore needed.

In the context of the Internet and with the communication protocol known as Hypertext Transfer Protocol, a content file is typically specified by an alpha numeric string known as a Uniform Resource Locator (URL). The URL includes an address portion which indicates a network address of a home server from which to obtain the desired information. The URL also includes a local information text stream that identifies the requested information which may, for example, be a file name, a search request, or other identification.

After the user specifies a URL to the client browser computer, the address portion of the URL is sent over the network. Once the connection with the server is established, the client can then retrieve a desired document by passing the local information text stream over the network to the home server. The home server then retrieves the document from its local disk

or memory storage and transmits the document over the network down to the client. The network connection between the home server and the client is then terminated.

However, an additional step is typically required in the context of streaming media files. For example, Microsoft has a set of multimedia technology tools for assisting with the creation of Web streaming media content. This tool set is an open, extensible platform that allows one to define a streaming media format file which makes it possible to combine a multitude of data types, such as audio, video, still images, URLs, etc. into an integrated data stream. One of the tools is a Microsoft Media Player tool that augments the capabilities of a Web browser to display multimedia content. Companies such as Apple Computer, RealNetworks, and Xing Technologies also provide standardized media players and streaming media file formats.

In the typical scenario, the user of a Web browser requests that a media stream file be downloaded, such as sending, in particular, the URL of a media redirection file from a Web server. The media redirection file (MRF) is a type of specialized Hypertext Markup Language (HTML) file that contains instructions for how to locate the multimedia file and in what format the multimedia file is in. The Web server returns the MRF file to the user's browser program. The browser program then reads the MRF file to determine the location of the media server containing one or more multimedia content files. The browser then launches the associated media player application program and passes the MRF file to it. The

media player reads the MRF file to obtain the information needed to open a connection to a media server, such as a URL, and the required protocol information, depending upon the type of medial content is in the file. The streaming media content file is then routed from the media server down to the user.

SUMMARY OF THE INVENTION

The problem with standard techniques for handling media redirection files is that the media server must have a connection open to it for each browser which needs access to the streaming media content.

The present invention avoids this difficulty by deploying a multimedia cache and a cache redirector functionality within the network. In particular, the invention locates a multimedia cache at a point of presence, such as at an Internet Service Provider (ISP), at an intranet gateway, or at other places close to the edge of the network near the user. A redirector function allows requests for a multimedia program content to be redirected to the local media cache server instead of routing them up through the network. This rerouting is transparent to the operation of the media server as well as being transparent to the user's browser and media player programs.

Now more particularly, in a preferred embodiment, a user's browser program first requests a media redirection file (MRF) from an origin Web server. The Web server then returns the MRF file to the user's browser program. As in the prior art, the MRF file contains instructions for telling the browser where to locate the streaming media content file and what type of player program will be needed to view it. However,

as the MRF file is returned from the media server, the cache redirector intercepts the MRF file before it is actually forwarded to the browser. The cache redirector examines the MRF file to determine if the requested multimedia content file is available in the local media cache. If so, then the designation in the MRF file is changed by the cache redirector to indicate the local media cache as the location of the requested media content. On a local cache miss operation, a request for the media file is launched to the indicated media server. In either event, the browser then opens the media player and the media player obtains the stream from the local media cache.

In the aforementioned arrangement, the cache redirector should maintain, or at least have access to, information concerning the contents of the local media cache. In particular, the cache redirector at a minimum has access to information as to what is contained in the media cache so that it can determine whether or not the media file being requested is already resident in the local media cache. The cache redirector may preferably also manage the media cache, telling it what content files to store and what to delete.

The media cache typically uses the same type of computing equipment as the media server, i.e., a high speed computer processor with high speed disk storage capable of managing connections to various network layers for a number of users. The media cache may support any number of different file formats such as Microsoft Media Technologies, Apple Computer, RealNetworks, Xing Technologies, or other types of streaming file formats.

The cache redirector can be a small network server (appliance) whose only function is media file redirection. The cache redirector watches packet traffic arriving at the point of presence, rewrites MRF files as needed, and keeps track of what is contained in the media cache. However, the cache redirector may also itself contain cache memory for caching other types of content such as text files and the like.

In another embodiment of the invention, the cache redirector does not need to keep track of what is in the media cache itself. In this instance, the cache redirector communicates in some way with the media cache in order to be able to query it as to the media cache. In this scenario, the media cache itself is intelligent about storing streaming media content files.

The advantage of the foregoing arrangement is especially realized in environments where high speed modem equipment, such as Digital Subscriber Line (xDSL) or cable modem equipment, is deployed between the point of presence and the end users. While such equipment can run at speeds of 500 kilobits per second or more, the Internet itself can sometimes only provide 40 kilobits per second or even less per user connection. Thus, by placing the media cache with the media file redirection protocol at the edge of the network closest to the user, such as at a point of presence, streaming media files can be presented to users at the highest available data rates provided by their high speed interface equipment, without having to tolerate delays through the Internet.

Many types of specialized content files such as banner ads, streaming video clips, and the like can

benefit from the deployment of a cache redirector and local cache server.

In an alternative embodiment, the media redirection file can instead be modified by the origin server. In this embodiment, the origin server has access to information concerning the address of media caches located at points of presence in the network. The net effect is that the cache redirector functionality is instead resident at the origin server, which modifies the MRF before it is passed to the user.

In yet another embodiment, a Directory Name Service (DNS) can be used to implement the cache redirection functionality. This embodiment may be preferred for networks where it is not possible to intercept traffic or modify origin servers. Specifically, a DNS for the network zone in question may be queried by the user when attempting to locate the MRF file or even the media file itself. In this embodiment, the DNS returns the address of the local media cache to the user.

It is also possible to pre-position media file content at the media cache servers. This helps expedite even the initial service of media files to users in a network zone for a given media cache.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the invention will be apparent from the following more particular description of preferred embodiments of the invention, as illustrated in the accompanying drawings in which like reference characters refer to the same parts throughout the different views. The drawings are not necessarily to

scale, emphasis instead being placed upon illustrating the principles of the invention.

Fig. 1 is a system diagram of a computer network in which media request file redirection may be implemented according to the invention.

Fig. 2 is an example of a media request file in which a redirection takes place.

Fig. 3 is a flow diagram of the steps performed by various components of the network.

Fig. 4 is a modified system diagram showing media caches distributed throughout the network.

Fig. 5 is a modified series of steps performed in the environment of Fig. 4.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

Turning attention now to the drawings more particularly, Fig. 1 is a system level diagram of a computer network in which media file requests are served according to the invention. As is typical in computer networks such as the Internet, a home or origin location 10 contains a number of computers. At certain points of presence (POPs) 30 in the network, additional computing and communications equipment permit the establishment of connections over the Internet 20 to other computers (known as clients) at a user site 40. The computers at the origin location 10 are typically referred to as servers. In the context of the Web, for example, one type of computer may be a Web server 12 that serves documents in the well known Hypertext Markup Language (HTML), or other formats. In this environment, the clients 40 run software that is enabled to interpret and display HTML files such as Web

browsers like Netscape Navigator, or Microsoft Internet Explorer.

In the illustrated scenario, a media server 14 contains multimedia documents such as streaming audio or streaming video files that are associated with the Web server 12. The media server 14 is typically implemented with an architecture especially optimized for supporting the high speed presentation of such video and audio files. Both the Web server 12 and media server 14 make connections through a router 16 to provide such media files to client computers 40 connected to the Internet 20. Although the Web server 12 and media server 14 are shown as being co-located at the same site, it should be understood that they may be physically located at different sites.

At the point of presence 30, computing and communications equipment include a router 32, a link 34, a cache redirector 36, a media cache 38, and a local area network 39. The point of presence 30 may, for example, be an intranet gateway or an Internet service provider (ISP). In the former case, the end nodes 42 may be directly connected to the local area network 39. In the latter case, a Remote Access Server (RAS) device 45 provides communication connections through another network 50 such as the Public Switch Telephone Network (PSTN) to subscribers units 52 located, for example, within homes or businesses that make connections through a dial up network.

The remote access server 45 preferably makes use of high speed subscriber interconnection equipment such as Digital Subscriber Line (xDSL), cable modems, or other high speed interconnectivity devices. The remote access server 45 may, for example, be a xDSL

multiplexer, cable modem head end, T1 converter, or other such equipment which concentrates Internet protocol (IP) type traffic and forwards it over connections such as the public switch telephone network or cable distribution network.

In accordance with the invention, also located at the point of presence 30 is a media cache 38. Media cache 38 is populated with media files that are most popularly requested by the users of the client computers 42 and 52. In general, the idea behind the present invention is that when a streaming media file request is made by one of the clients 42 or 52, steps are taken to ensure that the request is served from the media cache 38 located at the point of presence 30, as opposed to having to retrieve the requested streaming media file from the media server 14 on the other side of the Internet 20.

The present invention in particular operates in an environment where requests for media files are made in the form of redirection objects generated by the browser programs located at the clients 42, 52. In particular, the browser programs 54 located at the clients 42, 52 are typically not capable of directly displaying or playing streaming media files. In particular, the browser needs to open or "launch" an associated application such as a media player program. One example of such a media player program which is well known in the art is the RealPlayer™ software available from RealNetworks of Seattle, Washington, or the Media Player™ software available from Microsoft Corporation of Redmond, Washington. Thus, when a media file is presented to the browser, the browser, strictly speaking, does not handle the media file but rather

redirects the request for the file to an appropriate media player program 56.

As is known in the art, when one of the users of the client computers 42, 52 requests a media stream such as a video clip of a presidential news conference from a site such as www.cnn.com, the browser 54 makes a request for the media stream in the form of a file request using the hypertext transfer protocol (HTTP) for a media redirection file (MRF) from the Web server 12. The Web server 12 sends the MRF file down to the browser 54. The MRF file itself contains additional instructions for the type of media player 56 that is required to view the video clip, as well as instructions for where to locate the media stream itself. The requested media file may be a recorded clip or a live stream.

For example, turning attention to Fig. 2, there is shown one type of MRF file known as an Active Streaming Format (ASF) file formatted for a Microsoft Media Technologies type environment. As can be seen, the MRF file is similar to a HTML type file in that it is a tent file that contains tag portions. One of the tags contains an <href> type reference to a media file which can be obtained by using the "mms" protocol at the URL //wmt.streamvision.com/msns/clinton.28.asf. The <href> reference indicates that the desired streaming video file of President Clinton's news conference is in the ASF format and can be obtained at the indicated uniform resource locator (URL).

In the prior art, the MRF file is returned directly to the browser 54 which in turn causes the browser 54 to open the media player 56 program such as the associated Microsoft Media Player program at the

user node 42, 52. The media player 56 then obtains the streaming media file indicated by the MRF file from the media server 14 by submitting another file request over the Internet 20. The streaming media file is then sent from the media server 14 to the media player 56.

In accordance with a preferred embodiment of the invention, a different sequence of events occurs in order to maximize the speed at which the media file is made available to the clients 42, 52. As shown in Fig. 3, from an initial idle state 100, the first three steps occur as with the prior art. In particular, in state 102, the user makes a request for the streaming media file such as by specifying it at the browser 54 either by clicking on a hypertext link within a Web page, or by presenting the URL for the corresponding MRF file directly to the browser 54. In the next state 104, the browser program requests the MRF file from the Web server 12. In state 106, the Web server 12 then returns the MRF file through the Internet 20 in the direction of the client 42, 52.

However, in state 108, the link 34 and cache redirector 36 recognize and intercept the MRF file on its way through the Internet 20 towards the clients 42, 52. In a state 110, the cache redirector 36 thus determines whether or not a streaming file referenced by the MRF file is available in the local media cache 38. If it is, then processing continues to state 114. If it is not, however, then processing continues to a state 112 in which the redirector 36 sends a modified MRF request to the media server 14, requesting that the media file be instead sent to the media cache 38.

In any event, processing continues to a state 114 in which the redirector 36 changes the <href> reference

within the MRF file before forwarding the MRF file to the users 52, 42. This modification by the redirector 36 inserts the URL of the media cache 38 into the MRF file so that the request will be serviced by the media cache 38 instead of the media server 14.

In state 116, the browser opens the media player program 56 and in state 118 the media player 56 obtains the information stream from the media cache 38.

The cache redirector 36 must know or have access to information indicating what is stored in the media cache 38. The cache redirector 36 may therefore actually be responsible for controlling what is stored in the media cache 38 and so must manage storage availability in the media cache 38 so that old or unused files are overwritten or deleted.

The cache redirector 36 can be implemented as a small server that watches traffic and intercepts MRF files, and the reroutes them as necessary to the media cache 38. The media cache may be, for example, a type of computing system known as a Media Server available from Microsoft or a Real Video Server available from RealNetworks.

It should be understood that there may be a number of rules to determine whether or not the cache redirector 34 actually causes the media cache 38 to serve the requested content. For example, such may be done upon recognizing multiple requests for the same content file.

It can now be appreciated that the invention provides certain advantages to optimize Internet access of media files. For example, in most scenarios, the highest data rates available to one of the clients 42 or 52 that accessing information over the Internet 20

is on the order of 40 kilobits per second (kbps). However, access equipment supported by the remote access server 45 is now being deployed which provides data rates which are much greater. For example, Digital Subscriber Line (xDSL) type equipment or cable modem equipment is capable of rates of 500 kilobits per second or more. The full advantage of such equipment cannot be realized given that the Internet 20 itself is only capable of rates which are much slower. Therefore, by building the media cache 38 into points of presence 30, and implementing media file request redirection, then the full advantage of the available bandwidth of DSL or cable modem connections can be taken advantage of when delivering streaming media files to the end users 42, 52. A service provider who deploys the media cache 38 in this manner many therefore charge extra for this service.

It should be understood that other embodiments of the cache redirector 36 are possible. For example, it is not necessarily required that the cache redirector 36 maintain control over the contents of the media cache 38. In particular, a high speed connection between the cache redirector 36 and media cache 38 may provide the ability for the cache redirector 36 in state 110 to determine the contents of the media cache 38 by query. The benefit of this arrangement is that the cache redirector 36 itself does not need to be as smart as a cache controller and the intelligence in the media cache controller can be built into the media cache 38 itself. However, the difference with this arrangement is that communication between the cache redirector 36 and media cache 38 must take place as to the content of the media cache. In addition, there is

preferably some sort of latency built into the media cache 38 in this arrangement so that it when a query is made for particular content from the cache redirector 36, such locations are not overwritten for a period of time after such queries, with the assumption that the media content will shortly needed to be delivered to one of the clients 42, 52 after such a request.

The redirector function of step 114 may be physically deployed at the outer edge of the network such as at the point of presence 30 as shown. However, the redirector 36 may also be deployed at nodes within the network 20 itself, or even at the origin Web server 12.

In addition, a Directory Name Service (DNS) 24 is typically deployed in networks 20 such as the Internet to assist the browser 54 in resolving domain names to physical addresses. Thus, step 104 typically involves the browser requesting the DNS 24 to return an Internet Protocol (IP) layer address for the origin Web server 12 in the process of requesting the MRF file. The DNS 24 is again typically involved in step 116 when the modified MRF file is read by the browser to determine the location of the media content file.

Therefore, at either step 104 or step 116, the DNS 24 can, instead of returning the IP address of the Web server 12 or media server 14, be programmed to instead return the IP address of the local media cache 38. If this is done in step 104, the local media cache returns the MRF file to the user. If this is done at step 116, the local media cache 38 returns the requested media file.

Yet another variation takes advantage of the fact that the media caches 38 are mass storage devices.

This scenario permits the pushing of content media files out to the local media caches 38 in anticipation of the browser 54 actually requesting a file. This pushing will work as long as the address specified in the MRF associated with the media file is changed in synchronism with the storing of content in the media cache 38.

The media cache 38 may also serve different versions of the media file based upon measured link bandwidth between the clients 42 or 52 and the cache 38. Specifically, when an MRF file is intercepted by the cache redirector 36, a link bandwidth determination may be made for the particular client 42 or 52 which made the request. A different media file may then be returned to the client depending upon the observed bandwidth. Quality of Service (QOS) enforcement mechanisms may then be used to ensure that the measured bandwidth remains available for the duration of the session.

In a further scenario, it is possible to avoid rewriting of the MRF file by the cache redirector 36. In particular, after state 106 in which the Web server returns the MRF file, the MRF file need not be immediately intercepted by the cache redirector 36. Rather, the MRF file can be allowed to return directly to the clients 42, 52. The browsers at the clients 42, 52 can then return a request for a file located at the media server 14. However, at this point, the cache redirector 36 makes use of techniques such as the link layer redirection technique referred to in our co-pending patent application Serial No. 09/040,519 entitled "Message Redirector with Cut-Through Switch for Highly Reliable and Efficient Network Traffic

Processor Deployment," assigned to InfoLibria, Inc., the entire contents of which are hereby incorporated by reference. In this instance, the user's request to the top level media server 14 is instead intercepted by a link layer redirection capability of the cache redirector 36. Within the clients 42, 52, a port connection may be redirected or spoofed, such as at the link layer, and this capability routes any request to the media server 14 instead to the local media cache 38. The link layer redirection capability within the cache redirector 36 in this instance needs to keep track of the contents of the media cache 38 and media server 14.

In a scenario such as shown in Fig. 4, media caches 38 may be distributed throughout various places in the network. The media server 14 keeps track of the URLs for these media caches 38. In such an instance, the media server 14 may return an MRF file containing not only the single specification of media server shown in Fig. 2, but multiple <href> references to other media servers.

When the cache redirector 36 obtains such an MRF file with multiple media cache possibilities, as shown in the diagram of Fig. 5, from state 108 entering a state 120, it examines the MRF file to determine the closest media cache from the group of media caches 38 specified in the MRF file. In state 124, this enables the cache redirector 36 to obtain the requested streaming media file from the closest media cache 38 which it can access, avoiding access to the top level media server 14 as much as possible. The closest media cache may not necessarily be the physically closest but

rather the cache with fastest relative access time. A state 122 routes any miss request to the closest cache.

In this scenario, the media server 36, rather than keeping track of where the content is located, may also be implemented by having various media caches 38 communicate with one another as to their contents. Techniques such as load balancing can also be used to further efficiently distribute frequently requested content throughout the network 10. What is important is that a set of possible media caches are presented to the cache redirector 36 and the cache redirector 36 has a selection method from which to choose among the various media caches 38.

The multiple media caches 38, for example, may cooperate to provide fault tolerance, clustering, or other aspects of the domain naming system which take advantage of the multi-processor environment.

It should also be understood that still other embodiments are possible. For example, if a media player is integrated into the browser program, then no MRF file need necessarily be routed to the browser 54. In this instance, a direct request for multimedia files can still be intercepted at the appropriate port connection with link layer redirection as described in Fig. 4.

While this invention has been particularly shown and described with references to preferred embodiments thereof, it will be understood by those skilled in the art that various changes in form and details may be made therein without departing from the spirit and scope of the invention as defined by the appended claims.

CLAIMS

What is claimed is:

1. A method of servicing requests for delivery of a media content file in a network of client-server computing systems in which a client computer makes an inquiry to an origin server to locate a media server associated with the origin server which stores the media content file, and wherein a local media cache is located within the network at a point near the client, the method comprising the steps of:

at the client, requesting delivery of the media content file by requesting from the origin server the delivery of a media redirection file (MRF) containing a redirection object specifying instructions for obtaining the media content file from the media server; and

prior to delivery of the media redirection file to the client, intercepting the media redirection file and rewriting the instructions contained therein so that the media content file is obtained from the local media cache by the client instead of from the media server directly.

2. A method as in claim 1 wherein the step of intercepting the media redirection file is performed at a node in the network near the client.

3. A method as in claim 1 wherein the step of intercepting the media redirection file is performed at an intermediate node in the network.

4. A method as in claim 1 wherein the step of intercepting the media redirection file is performed at the origin server.

5. A method as in claim 1 wherein the step of intercepting the media redirection file is modified at a link layer.

6. A method as in claim 1 wherein the local media cache is selected from among multiple media caches.

7. A method as in claim 1 wherein the media content file is stored in the local media cache prior to the client requesting delivery of the media content file.

8. A method as in claim 1 additionally comprising the step of:

determining an observed link bandwidth for file transfers between the local media cache and the client; and

rewriting the instructions in the media redirection file to specify one of a plurality of media content files depending upon the observed link bandwidth.

9. A method as in claim 6 wherein the multiple media caches are arranged and selected in accordance with a fault tolerance capability.

10. A method as in claim 1 wherein the multiple media caches are arranged as a cache cluster.

11. A method as in claim 1 additionally comprising the step of:

charging the user of the client computer a premium for intercepting media redirection files.

12. A method as in claim 1 additionally comprising the step of:

before rewriting the instructions contained in the media redirection file, determining if the media content file is stored at the local cache server.

13. A method as in claim 1 wherein the step of rewriting the instructions further comprises the step of rewriting an <href> tag within the media redirection file to insert a Uniform Resource Location (URL) of the media cache.

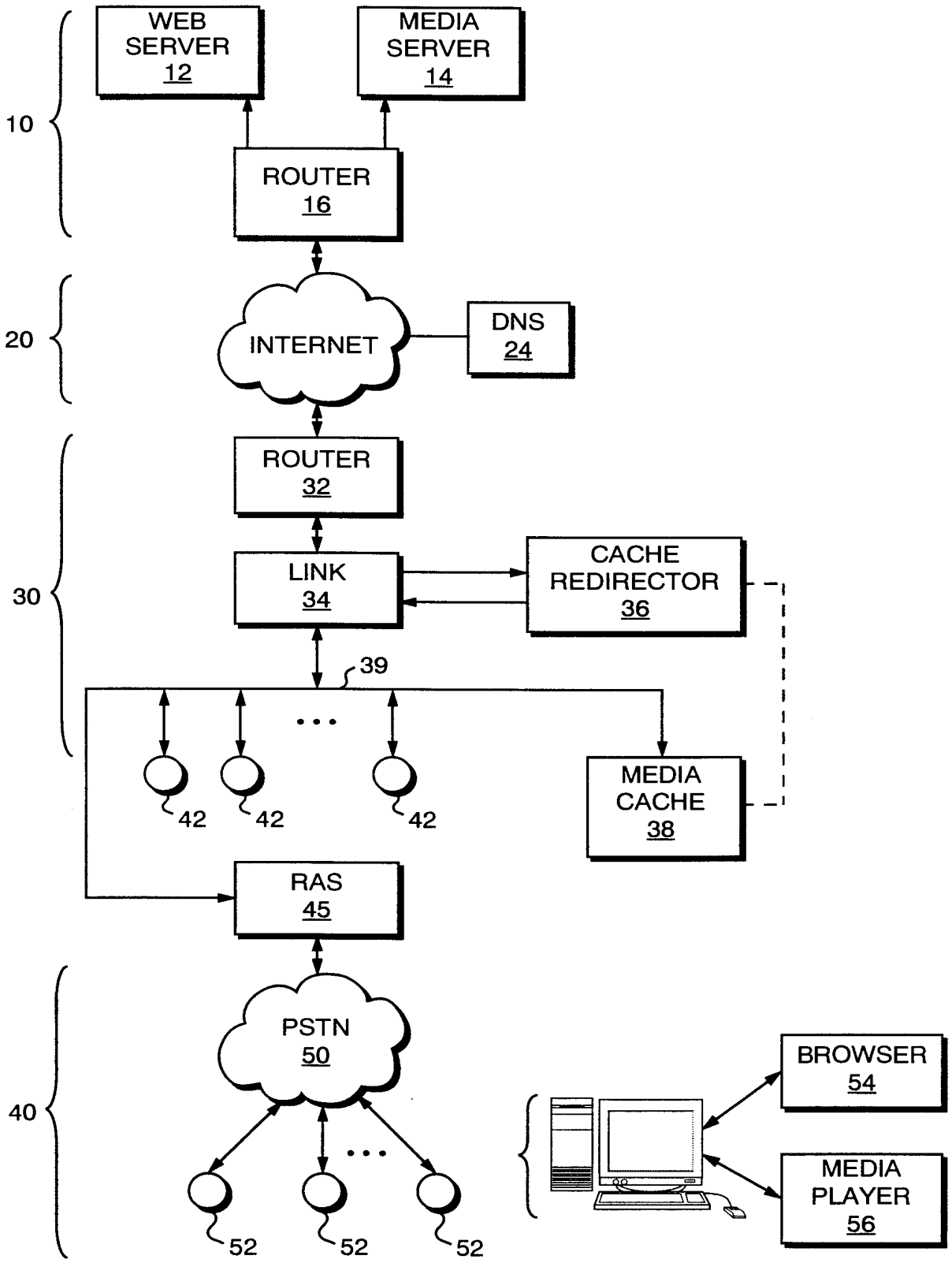


FIG. 1

SUBSTITUTE SHEET (RULE26)

```
Here is a sample asx file; notice the redirection

<ASX VERSION="3">
<Title>Video 17Kbps/Audio 5Kbps</TITLE>
<AUTHOR>Microsoft Windows Media Technologies</AUTHOR>
<ABSTRACT>30fps Streaming Video!</ABSTRACT>

<ENTRY>

  <ref href="mms://wmt.streamvision.com/msns/Clinton28.asf" />
  <TITLE>Press Conference</TITLE>
  <AUTHOR>Video Courtesy of CNN</AUTHOR>
  <COPYRIGHT>? 1998 Cable News Network, Inc.</COPYRIGHT>
  <MOREINFO href="http://www.cnn.com" />
  <ABSTRACT>President Clinton Press Conference</ABSTRACT>

<ENTRY>

</ASX>
```

FIG. 2

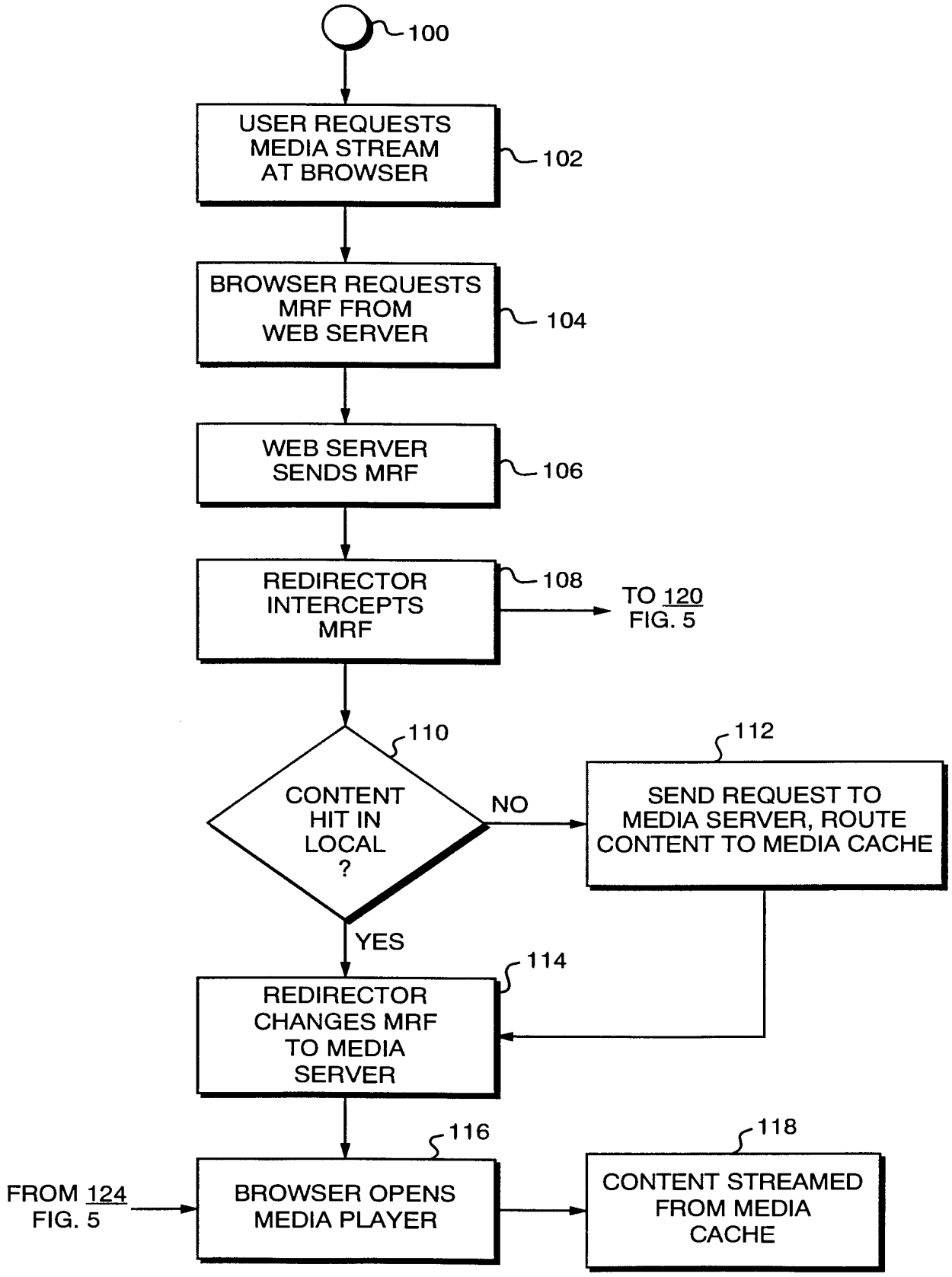

100
USER REQUESTS MEDIA STREAM AT BROWSER
102
BROWSER REQUESTS MRF FROM WEB SERVER
104
WEB SERVER SENDS MRF
106
REDIRECTOR INTERCEPTS MRF
108
TO 120 FIG. 5
110
CONTENT HIT IN LOCAL ?
NO
112
SEND REQUEST TO MEDIA SERVER, ROUTE CONTENT TO MEDIA CACHE
YES
114
REDIRECTOR CHANGES MRF TO MEDIA SERVER
116
FROM 124 FIG. 5
BROWSER OPENS MEDIA PLAYER
118
CONTENT STREAMED FROM MEDIA CACHE

FIG. 3

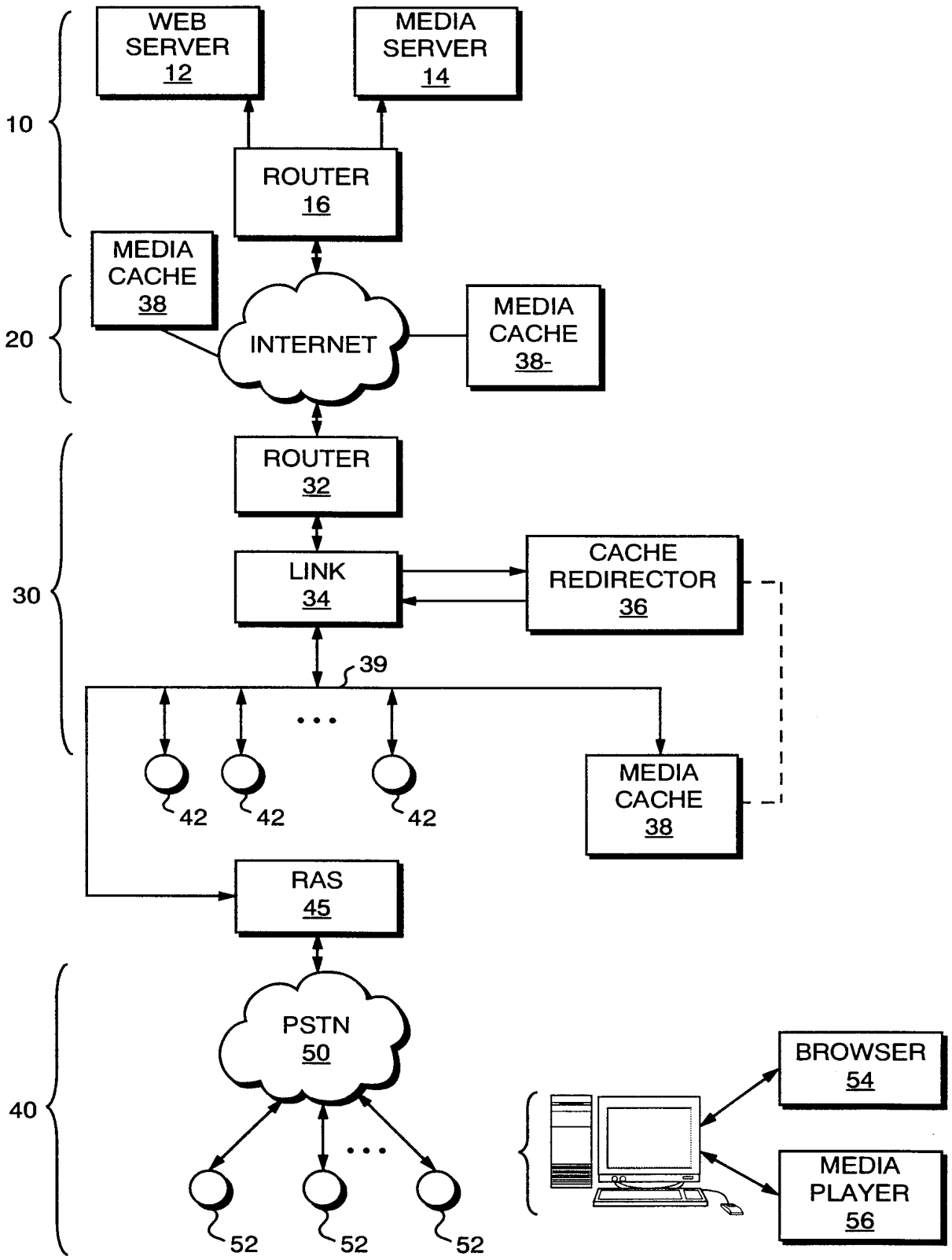

WEB SERVER 12
MEDIA SERVER 14
10
ROUTER 16
MEDIA CACHE 38
20
INTERNET
MEDIA CACHE 38-
ROUTER 32
LINK 34
CACHE REDIRECTOR 36
30
39
42
42
42
MEDIA CACHE 38
RAS 45
PSTN 50
40
52
52
52
BROWSER 54
MEDIA PLAYER 56

FIG. 4

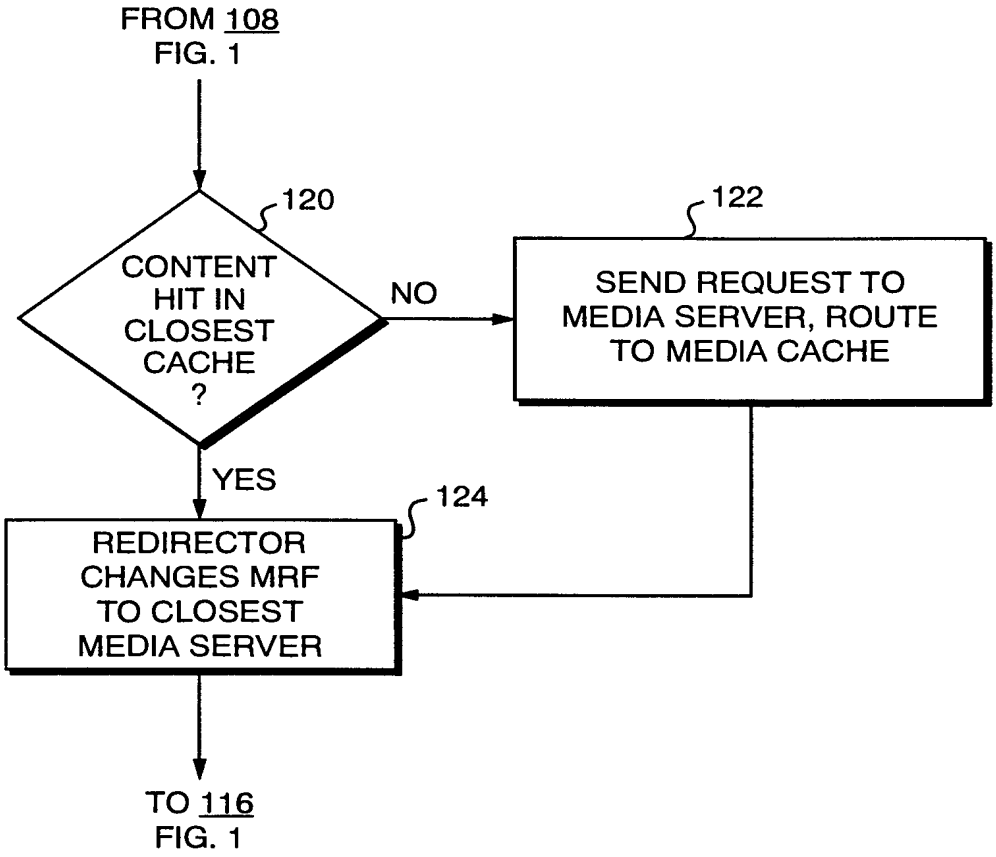

FROM 108
FIG. 1
120
CONTENT HIT IN CLOSEST CACHE ?
NO
122
SEND REQUEST TO MEDIA SERVER, ROUTE TO MEDIA CACHE
YES
124
REDIRECTOR CHANGES MRF TO CLOSEST MEDIA SERVER
TO 116
FIG. 1

FIG. 5

INTERNATIONAL SEARCH REPORT

national Application No
PCT/US 99/27325

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F17/30 H04L29/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 98 24208 A (JONES RICHARD ;ZANCANI LEO (GB); DONKIN RICHARD (GB); LAURIE BEN () 4 June 1998 (1998-06-04) page 2, line 30 -page 3, line 23 page 16, line 25 -page 18, line 24 page 24, line 1 - line 24 page 32, line 32 - line 34 --- -/--	1-8, 11-13

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
5 April 2000	12/04/2000

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Fournier, C

1

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	DE BRA P M E ET AL: "Information retrieval in the World-Wide Web: Making client-based searching feasible" COMPUTER NETWORKS AND ISDN SYSTEMS,NL,NORTH HOLLAND PUBLISHING. AMSTERDAM, vol. 27, no. 2, 1 November 1994 (1994-11-01), pages 183-192, XP004037989 ISSN: 0169-7552 page 188, right-hand column, paragraph 3.1.4 - paragraph 3.2	1-4,7, 12,13
A	MOURAD A ET AL: "SCALABLE WEB SERVER ARCHITECTURES" PROCEEDINGS IEEE SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS, 1 July 1997 (1997-07-01), XP000199852 the whole document	1

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

	national Application No
	PCT/US 99/27325

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9824208 A	04-06-1998	AU 5062698 A	22-06-1998
		EP 0940024 A	08-09-1999
		GB 2319710 A,B	27-05-1998

Form PCT/ISA/210 (patent family annex) (July 1992)

PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : H04N 7/10	A1	(11) International Publication Number: **WO 00/60861**
		(43) International Publication Date: 12 October 2000 (12.10.00)

(21) International Application Number: PCT/US00/08402

(22) International Filing Date: 30 March 2000 (30.03.00)

(30) Priority Data:

60/127,396	1 April 1999 (01.04.99)	US
09/447,472	23 November 1999 (23.11.99)	US

(71) Applicant: DIVA SYSTEMS CORPORATION [US/US]; 800 Saginaw Drive, Redwood City, CA 94063 (US).

(72) Inventors: ARMSTRONG, James, B.; 156 Nassau Street, Princeton, NJ 08540 (US). GOODE, Christopher, W., B.; 722 Creek Drive, Menlo Park, CA 94025 (US).

(74) Agents: MOSER, Raymond, R. et al.; Thomason, Moser and Patterson LLP, 2–40 Bridge Avenue, P.O. Box 8160, Red Bank, NJ 07701 (US).

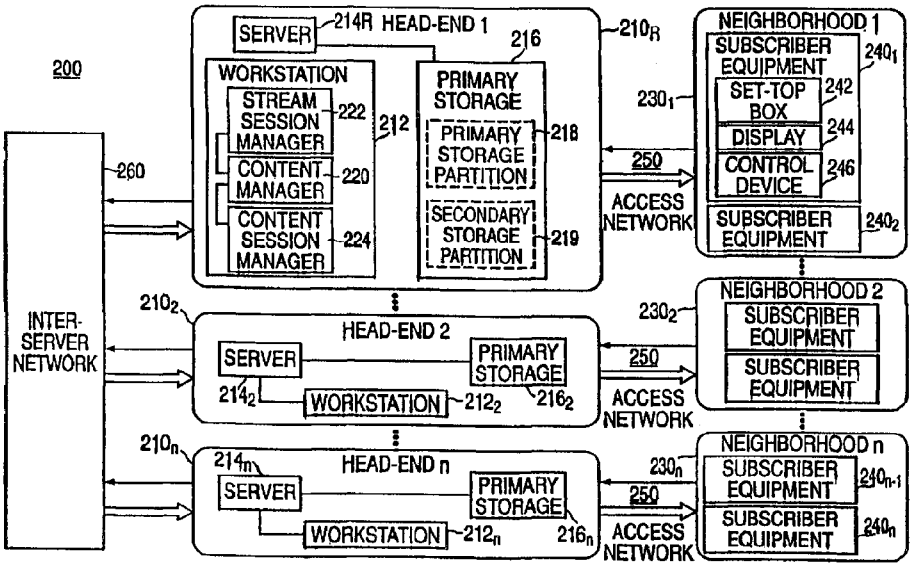
(81) Designated States: AE, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CU, CZ, DE, DK, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, UA, UG, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published

With international search report.

Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: METHOD AND APPARATUS FOR HIERARCHICAL DISTRIBUTION OF VIDEO CONTENT FOR AN INTERACTIVE INFORMATION DISTRIBUTION SYSTEM



(57) Abstract

A method (400, 500) and apparatus (200) for storing and distributing video information in an interactive information distribution system (200) by storing video information on either a centralized or de–centralized secondary storage partition (219, 119R) in a network of provider and subscriber equipment (140), and therein distributing the video information to a subscriber having made a request for such video information.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

METHOD AND APPARATUS FOR HIERARCHICAL DISTRIBUTION OF VIDEO CONTENT FOR AN INTERACTIVE INFORMATION DISTRIBUTION SYSTEM

CROSS REFERENCE TO RELATED APPLICATIONS

This application claims benefit of U.S. Provisional Application No. 60/127,396, filed April 1, 1999, which is hereby incorporated by reference in its entirety.

BACKGROUND OF THE DISCLOSURE

1. Field of the Invention

The present invention relates to a method and apparatus for providing an improved content delivery system for use with an interactive information distribution system. More particularly, the invention relates to an improved system for allocating, storing and distributing information within a video on demand (VOD) system.

2. Description of the Background Art

Recent advances in digital signal processing techniques and, in particular, improvements in digital compression techniques, have led to a plethora of proposals for providing new digital services via existing telephone, coaxial cable and satellite data networks. For example, it has been proposed to provide hundreds of cable television channels to subscribers by compressing digital data and digital video, transmitting the compressed digital signals over conventional coaxial cable television channels, and then decompressing the signals in the subscriber's receiver.

Another application for this technology is a video-on-demand (VOD) system in which a subscriber communicates directly with a video service provider via telephone lines to request a video program from a video library. The requested video program is then routed to the subscriber's home via telephone lines or via coaxial cable television cables for immediate viewing.

The VOD system may use a frequency multiplexing technique to enable control information for a subscriber's receiver to be transmitted through a cable network back to an information server. Such a system permits bi-directional

communications over a single network. In addition, graphical menus are displayed upon the subscriber's television and by using a remote control device, a subscriber selects a desired program for viewing.

Many VOD distribution systems established by the service providers utilize a plurality of cable "head-ends", where each head-end serves as a distribution point for a respective "neighborhood" of subscribers.

Each head-end comprises one or more VOD server systems. Each VOD server system contains subsystems that interface with each other to manage, store and distribute video content. The subsystems usually are composed of a video stream server, primary and secondary data storage units, and host workstations on which various management software applications execute.

The subscriber has thousands of video titles from which to choose. A subscriber selects a video title displayed on their monitor and a signal is sent from the subscriber's location to the local head-end server. The software management applications, for each of the local video stream servers, queries a database to inquire if the chosen content is presently stored at the head-end storage devices. The most popular video content is typically stored locally on the primary storage device. The secondary storage device is typically a large library of the remaining video content.

If the content is present at the head-end, the server will deliver the video package to the subscriber. Alternatively, if neither storage devices at the head-end have the selected video information, the head-end must first retrieve the requested content from the service provider before delivery. Because of this deficiency, there is a delay in transmission of the video information to the subscriber.

Head-ends that are networked together are known as regions and as such, are served by several VOD systems. The service provider, through a managerial software application, controls the introduction and removal of video information to and from these servers. Thus, each region has a plurality of servers and storage equipment, with each head-end having duplicate video content stored therein.

All of this additional storage capacity, in the form of additional storage hardware results in higher system costs. These costs are primarily due to procuring and maintaining reliable storage devices. This in turn increases the cost of building infrastructure and subsequently, the ultimate cost to the consumer.

Thus, there is a need to reduce the cost of implementing and maintaining replicated head-end located storage libraries, while still maintaining the quality and quantity of video titles offered by the service provider. To achieve this, there is a need to support low latency file transfer operations at relatively high but variable rates using a variable bit rate (VBR) transport, support high quality streaming transport at various control bit rates, as well as supporting appropriate control message flows.

SUMMARY OF THE INVENTION

The disadvantages heretofore associated with the prior art are overcome by the present invention of a system for storing and distributing video information in an interactive information distribution system. In the exemplary embodiment of the invention, a system head-end is comprised of subsystems that are used to manage, store and distribute video content: namely a video stream server, a primary storage device and a management application that is executable through a general purpose computer (e.g., a workstation host or personal computer).

The system redistributes the secondary storage devices to serve as a shared secondary storage partition on an inter-server network. As such, various head-end servers can share information, thus reducing the overall storage requirements for the system. Such a beneficial reduction in storage hardware can be made because of a distributive method of allocating the video libraries.

The inventors have recognized that of the vast number of available video titles (and associated video assets), only a small percentage of titles may be considered "popular" by the subscribers at any one time. Therefore, a large portion of the video assets are not requested with any frequency. For example, if there are 1000 movies, typically less than 25% will actually be requested on a regular basis to qualify as being popular. This means that the remaining 750 movies are infrequently requested and are being stored at each head-end without being accessed very often. Such under-utilization and replication of these storage assets is costly to sustain.

A method of monitoring and storing video information, based upon a rate of video requests from subscribers, enables the system to store the content more efficiently. If shown to be in popular demand by the consumer, such as by

surpassing a threshold request rate set by a service provider administrator, the "high request" video assets are stored across all the neighborhood servers. The remaining "low request" video assets are divided and distributed among the secondary storage partitions. Numerous threshold rates may be established for each video asset. This method allows multiple disbursements of the video asset to either a single video server or some multiple thereof, which is less than all of the video servers in the entire VOD system.

In one embodiment of the invention, a "remote server" is advantageously configured to replace all of the secondary storage devices located at each of the head-ends. The remote server and its associated primary and secondary storage partitions are established at a remote site and linked within a network of preferably three or more head-ends. These storage partitions contain the library of VOD assets presently stored on both thc primary and secondary storage devices. The primary storage partition holds frequently requested video assets while the secondary storage partition (e.g. a magneto optical storage device, fiber channel hard drives or otherwise) holds the remaining infrequently requested video content. To manage the distribution of video assets, the remote server executes various management tools.

In those instances when a subscriber has requested video assets not stored on the primary storage partition at the local head-end, the content manager will retrieve the desired video content from the remote server. The remote server may function as a head-end itself by serving the video assets directly to the subscribers. This is accomplished by transmitting the video assets through an access network at the subscriber's local head-end and thereby bypassing the local server. Alternatively, the remote server may be used as a source for copying video assets to other head-end servers for storage and distribution.

In a second embodiment, the network of local stream servers is considered as a "virtual remote server." This extends the concept of a remote server as the central repository of all video assets in the network. Using this method, a number of stream servers connected by a fast network capable of constant and variable bit rate (CBR and VBR) transport service, will provide a subscriber with a selection of movies and other assets that are greater than the storage capacity of any individual stream server.

The video assets considered popular, as determined by the frequency of subscriber requests, are stored on the primary storage partitions in all the neighborhoods. The remaining infrequently requested video assets are divided and then selectively distributed amongst the neighborhood head-ends on the secondary storage partitions.

One function that this VOD network satisfies is to transport video streams at various bit rates with an appropriate quality of servers. Another function is to support low latency file transfer operations at relatively high bit variable rates, using a VBR transport. Finally, the network supports appropriate control message flows.

Thus, the current requirement of having secondary storage partitions for additional storage of infrequently requested video content, as well as having backup at each head-end site has been eliminated. This redistribution of storage space by either embodiment provides geographical locality for the introduction and removal of video content. Similarly, it reduces the cost for storing archived content, improves performance in retrieving content, and reduces the disk storage capacity of the individual stream servers.

BRIEF DESCRIPTION OF THE DRAWINGS

The teachings of the present invention can be readily understood by considering the following detailed description in conjunction with the accompanying drawings, in which:

FIG. 1 depicts a high-level block diagram of a first embodiment of an interactive information distribution system in accordance with the invention;

FIG.2 depicts a high-level block diagram of a second embodiment of an interactive information distribution in accordance with the invention;

FIG.3 depicts a graphical representation of the interrelationships amongst a plurality of management tools, provider equipment and subscriber equipment useful in understanding the invention;

FIGS.4A and 4B together depict a flow diagram of a method of allocating video information suitable for use in the interactive information distribution systems of FIG. 1 and FIG. 2; and

FIG.5 depicts a flow diagram of a method of storing video assets at various locations suitable for use in the interactive information distribution systems of FIG. 1 and FIG. 2..

To facilitate understanding, identical reference numerals have been used, where possible, to designate identical elements that are common to the figures.

DETAILED DESCRIPTION OF THE INVENTION

In an interactive information distribution system, a method and apparatus for distributing video assets to a plurality of subscribers, and redistributing video assets stored on primary and secondary storage devices. FIG. 1 depicts a high-level block diagram of an interactive information distribution system.

Specifically, the interactive information distribution system 100 comprises a plurality of head-ends 110_1 through 110_n (collectively head-ends 110), each of said head ends 110 being coupled to at least one of a corresponding plurality of neighborhoods 130_1 through 130_n (collectively neighborhoods 130), and a remote server head-end 110R. Each of the head-ends 110 comprises various subsystems that store and distribute video content. The terminology "video content", "video asset" and "video information" as used herein are equivalent.

Moreover, each head-end 110 comprises a host workstation 112, a video stream server 114 and a primary storage partition 118. The primary storage partition 118 and host workstation 112 are coupled to the video stream server 114. In the exemplary embodiment, the primary storage partition 118 comprises of an array of small computer systems interface (SCSI) or fiber channel hard drives.

The host workstation 112 allows an operator to interface with a series of management tools including a content manager 120, a stream session manager 122 and content session manager 124. These management tools perform various system command and control functions such as allocating transmission paths and monitoring the video streams through a communications network to and from the neighborhoods 130. The managers may be implemented as hardware, software, or a combination of hardware and software.

Each neighborhood 130 serves a plurality of subscribers. The interactive information distribution system 100 transmits video streams from the video stream

server 114 to the subscriber's respective subscriber equipment 140. The subscriber equipment 140 comprises a set-top box 142, a display 144 and a control device 146.

The neighborhoods 130 are linked to their respective head-ends 110 through an access network 150. The stream session manager 122 at each of the head-ends 110, controls the transfer of video assets over the access network 150. Throughout this description the term "local" will be used to denote a head-end 110 or subscriber equipment 140 that is associated with a neighborhood 130 in which a subscriber request for a video asset has been initiated.

The remote server head-end 110R comprises a remote server 114R (i.e., a video stream server) and a primary storage partition 118R coupled to the remote server 114R. A secondary storage partition 119R is also coupled to the remote server 114R. The secondary storage partition 119R illustratively may be magneto optical storage devices or hard disk drives. The host workstation 112R, which includes the content manager 120R, stream session manager 122R and content session manager 124R, interfaces with the plurality of head-ends 110, through an inter-server network 160.

A primary storage partition 118 of a head-end 110, including the remote server head-end 110R, is used to store frequently requested video assets. Alternately, the secondary storage partition 119R of the remote server head-end 110R is used to store infrequently requested video assets.

The content manager 120 tracks the number of requests for a video asset and produces an asset request rate. An operator using the host workstation 112 defines a threshold rate for each video asset. The content manager 120 periodically compares the asset request rate against the threshold rate for each video asset in the system 100. If the asset request rate traverse the threshold rate for a video asset, then the video asset is stored on the primary storage partitions 118 and 118R at each of the head-ends 110 and 110R. If the asset request rate does not traverse the threshold rate for a video asset, then the video asset is stored on the secondary storage partition 119R at the remote server head-end 110R. In this manner video assets are dynamically distributed throughout the interactive information distribution system 100.

A subscriber in a neighborhood 130 uses the control device 146 to select and request video assets for delivery and viewing on the display 144. At the local head-

end 110, the content manager 120 determines whether the video asset is locally stored at the head-end 110. If the requested video content is locally stored (i.e. on the primary storage partition 118), the local video stream server 114 accesses and delivers the locally stored video asset to the set-top box 142 of the requesting subscriber's equipment 140, thereby fulfilling the request.

If the video asset is not locally stored at the head-end 110, then the content manager 120 contacts the content session manager 124 at the remote server head-end 110R. The remote server head-end 110R will then transmit the requested video information by alternate methods, depending on the circumstances.

The decision as to whether to stream the requested video directly to the subscriber 140 from the remote head-end 110R, or transfer or migrate assets from the remote head-end 110R to the local head-end 110, and then stream from the local head-end 110 to the subscribers 140 depends on various factors. These include the availability of isochronous bandwidth on the inter-server network 160, the current usage of the requested movie or video asset by other subscribers, the quality of service selected for the delivery of the service, and the availability of space on the local video stream server 114 for storage of the video information. By using various algorithms, these factors are addressed by the content managers, which is generally responsible for checking the status of the servers.

One method comprises the steps of retrieving the requested video information from the secondary storage partition 119R of the remote server head-end 110R, and then the remote server 114R transmits the requested video through the inter-server network 160 to the local neighborhood 130 where the request originated. The video stream server 114 at the local neighborhood head-end 130 receives the requested video asset, and then transmits it to the requesting subscriber over the access network 150 for viewing on the subscriber equipment 140. This method may be used when it is also desirable to store the video asset at the local head-ends 110.

Alternately, a second method comprises the steps of the remote server 114R retrieving the requested video asset from the secondary storage partition 119R at the remote server head-end 110R. The remote server 114R then transmits the video asset over the inter-server network 160 and access network 150 directly to the requesting subscriber's subscriber equipment 140. In this way, the remote server

head-end 110R utilizes the local head-end 110 as a conduit for transmitting the video information and avoids interfacing with the local server 114.

FIG. 2 depicts the block diagram of a second embodiment of an interactive video distribution system 200. Specifically, the interactive information distribution system 200 comprises a plurality of head-ends 210_1 through 210_n (collectively head-ends 210), each of said head ends 210 being coupled to at least one of a corresponding plurality of neighborhoods 230_1 through 230_n (collectively neighborhoods 230).

The head-ends 210 are comprised of various subsystems that store and distribute video content. Specifically, a host workstation 212, a video stream server 214 and a primary storage device 216. The primary storage device 216 and host workstation 212 are coupled to the video stream server 214. In the exemplary embodiment, the primary storage device 216 comprises an array of small computer systems interface (SCSI) or fiber channel hard drives. Furthermore, the primary storage device 216 is apportioned into at least two storage partitions designated as a primary storage partition 218, and a secondary storage partition 219.

The host workstation 212 interfaces with a content manager 220, a stream session manager 222 and a content session manager 224. These are management tools that perform various system command and control functions, as well as allocate transmission paths and monitor the video streams through a communications network to and from the neighborhoods 230. The managers may be implemented as hardware, software, or a combination of hardware and software.

Each neighborhood 230 serves a plurality of subscribers. The interactive information distribution system 200 transmits video streams from the video stream server 214 to the subscriber's respective subscriber equipment 240. The subscriber equipment 240 comprises a set-top box 242, a display 244 and a control device 246.

The neighborhoods 230 are linked to their respective local head-ends 210 through an access network 250. The stream session manager 222 at each head-end 210 controls the transfer of video information to and from the subscriber equipment 240 over the access network 250. Throughout this description the term "local" head-end will be used to denote a head-end 210 that is associated with a neighborhood 230 or subscriber equipment 240 within that neighborhood 230 where a request for video information by a subscriber has originated.

The primary storage partition 218 on the primary storage device 216 at each head-end 210 is used to store frequently requested video assets and temporarily cached library video assets. Each primary storage partition 218 at each head-end 210 typically has the same frequently requested video assets as any other head-end 210.

The secondary storage partition 219 is used to store portions of the infrequently requested video assets. An entire library of infrequently requested video assets is divided and stored amongst the plurality of head-ends 210 at each of the secondary storage partitions 219 on their respective primary storage devices 216.

An infrequently requested video asset is typically stored on the secondary storage partition 219 at a single head-end 210. However, the request rate for that video asset may warrant additional storage at other head-ends 210. As such, the content may be replicated and stored thereafter. In this manner, video assets that do not warrant storage across the entire system of head-ends 210 in the interactive information distribution system 200, may still be dynamically stored at multiple head-ends 210. Such dynamic storage corresponding to those neighborhoods having higher request rates than others is made in accordance with an algorithm that allows maximum access to the video titles with minimum network cost associated with their delivery.

A subscriber in a neighborhood 230 uses a control device 246 to select and request a video asset for delivery and viewing on the display 244. At the local head-end 210, the content manager 220 determines whether the video asset is locally stored at the head-end 210. If the requested asset is locally stored (i.e. at the primary storage partition 218 on the primary storage device 216), the local video stream server 214 accesses and delivers the locally stored video asset to the requesting subscriber's equipment 240, thereby fulfilling the request.

If the video asset is not locally stored at the head-end 210, then the content manager 220 contacts the content session manager 224 at a remote head-end 210 where the infrequently requested video asset is located. Throughout this description the term "remote" head-end will be used to denote a head-end and its corresponding sub-systems that are associated with that head-end containing video information not locally available on at a local head-end when requested by a subscriber. The suffix

"R" will denote the corresponding elements at that head-end that form a portion of the remote head-end.

The video stream server 214R at the remote head-end 210R retrieves the infrequently requested video assets from its respective secondary storage partition 219R. The video stream server 214R then transmits the retrieved video information over the inter-server network 260 to the local head-end 210 from where the request for video information initiated (for example, head-end 210_1). The local video stream server 214, operating as an intervening server with respect to the requesting subscriber and remote head-end 210R, directs the requested video assets from the inter-server network 260 to the subscriber equipment 240, via the local head-end 210 and the access network 250.

Various factors may be involved as to whether to stream directly from the remote head-end 210R to the subscriber equipment 240. Typically, they include the availability of isochronous bandwidth on the inter-server network 260, the current usage of the requested movie or video asset by other subscribers, the quality of service selected for the delivery of the service, and the availability of space on the local video stream server 214 for storage of the video information.

Alternately, in the instance where local storage is desired, the video stream server 214 at the local head-end 210 receives the video information from the remote head-end 210R. Upon receiving the streamed information, the video stream server 214 concurrently stores the assets locally, and transmits the infrequently requested video information over the access network 250 to the requesting subscriber for viewing on the subscriber equipment 240.

The video assets may be streamed in either real time or in bursts. Where the video asset is simply being presented to the subscriber for viewing, the content is streamed in real time, thereby alleviating any need for buffering of the content. If the video asset is also going to be stored at the local head-end 210, then the video asset may be transmitted or migrated in bursts, i.e., streaming at a maximum rate of transfer by the remote video stream server 214R according to bandwidth availability. The video asset is then stored on the primary storage device 216 at the local head-end 210. After a sufficient amount of the video asset is stored, the local video stream server 214 will stream the video asset to the subscriber in real time for

presentation on the subscriber equipment 240. As the asset is sent to the subscriber, the remaining amount of the asset is stored in the primary storage device 216.

In the case where infrequently requested video content is deemed desirable to have stored at other head-ends 210, the infrequently requested video asset is retrieved from the remote head-end 210R having such video asset stored on the head-end's secondary storage partition 219. The infrequently requested video asset is then transmitted by the remote video stream server 214R in bursts across the inter-server network 260, to the local head-ends 210 designated by the content manager 220 to store such video asset.

Such movement of video assets occurs in this manner, based upon the system level algorithms having specified threshold rates, as opposed to an individual subscriber requesting a video asset stored at a remote head-end site. Thus, the video assets designated for movement from one server to another occur during system level maintenance, i.e., in the background to any subscriber requesting video assets.

Once each local video stream server 214 receives the infrequently requested video asset, it then stores it on its respective primary storage device 216 at the primary storage partition 218. In this manner, an infrequently requested video asset that later in time becomes frequently requested, may be removed from the remote secondary storage partition 219 at the remote head-end 210R, and transferred to some or all of the plurality of local head-ends 210. Thus, highly requested video assets are distributed broadly across the network of video stream servers. This serves as an effective method to satisfy the high request rates by the subscribers.

Conversely, a frequently requested video asset stored at the primary storage partition 218 on the primary storage device 216 at each head-end 210 may become infrequently requested over a period of time, or for that matter, during various hours of a day. The content manager 220 determines which head-ends 210 are to be designated as a remote head-end 210R for storing such infrequently requested video asset. In this instance, the content manager 220 also controls the transfer of that video asset from a local head-end 210 storing the video asset on its primary storage partition 218, to the secondary storage partition 219 of a head-end 210R designated to store such infrequently requested video asset. After the transfer of the video asset is made to the secondary storage partition 219, the video asset may be deleted from

the primary storage partitions 218 at each of the local head-ends 210 where the low request rate does not warrant duplicate storage.

These methods of asset allocation ensure greater storage capacity at each head-end 210 for content that is accessed by subscribers. Additionally, by distributing the video assets at selected local head-ends 210 based upon the subscriber request rates, network bandwidth will be used with greater efficiency and the subscribers will receive their product with minimal delay.

FIG.3 depicts a graphical representation of the interrelationships amongst a plurality of management tools, provider equipment, and subscriber equipment. Furthermore, FIG. 3 represents the subject matter of FIG. 1 only, however, FIG. 3 similarly applies to the subject matter depicted in FIG. 2. The content manager 120 and stream session manager 122 are located at each of a plurality of local head-ends 110, and are linked to each other. The stream session manager 122 manages and controls transmissions to and from the subscriber equipment 140 and a local video stream server 114 at the local head-ends 110 supporting the subscriber equipment 140.

An access network 150 is a transmission medium between the head-end 110 and subscriber equipment. The access network 150 may be comprised of telephone lines, coax cable, satellite links, fiber optics hybrid-fiber coaxial or a combination thereof.

A content session manager 124 is linked to the content manager 120, as well as communicating with a remote head-end 110R having a remote server 114R and storage partitions. The content session manager 124 controls the flow of video assets between the local video stream servers 114 at the local head-ends 110 and the remote head-end 110R. For purposes of this FIG.3, the use of the suffix "R" designates a remote location and hardware having infrequently requested video assets stored therein.

Illustratively, in the first embodiment as depicted by FIG.1, infrequently requested video assets are centrally stored on the secondary storage partition 119 at the remote server head-end 110R. Thus, the content session manager 124 is also centrally located at the remote server head-end 110R. In the second embodiment, as shown in FIG.2, the infrequently requested video assets are de-centrally located because they are divided amongst each of the plurality of head-ends 210 within the interactive video distribution system 200. Thus, the content session manager 224 is