Doc Code: TRACK1.REQ
Document Description: TrackOne Request

PTO/SB/424 (12-11)

CERTIFICATION AND REQUEST FOR PRIORITIZED EXAMINATION UNDER 37 CFR 1.102(e) (Page 1 of 1)

First Named Inventor:	DAVID F. BRUECK	Nonprovisional Application Number (if known):	Herewith
Title of Invention:	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING		

APPLICANT HEREBY CERTIFIES THE FOLLOWING AND REQUESTS PRIORITIZED EXAMINATION FOR THE ABOVE-IDENTIFIED APPLICATION.

1. The processing fee set forth in 37 CFR 1.17(i), the prioritized examination fee set forth in 37 CFR 1.17(c), and if not already paid, the publication fee set forth in 37 CFR 1.18(d) have been filed with the request. The basic filing fee, search fee, examination fee, and any required excess claims and application size fees are filed with the request or have been already been paid.

2. The application contains or is amended to contain no more than four independent claims and no more than thirty total claims, and no multiple dependent claims.

3. The applicable box is checked below:

I. ☑ Original Application (Track One) - Prioritized Examination under § 1.102(e)(1)

i. (a) The application is an original nonprovisional utility application filed under 35 U.S.C. 111(a). This certification and request is being filed with the utility application via EFS-Web.

---OR---

(b) The application is an original nonprovisional plant application filed under 35 U.S.C. 111(a). This certification and request is being filed with the plant application in paper.

ii. An executed oath or declaration under 37 CFR 1.63 is filed with the application.

II. ☐ Request for Continued Examination - Prioritized Examination under § 1.102(e)(2)

i. A request for continued examination has been filed with, or prior to, this form.
ii. If the application is a utility application, this certification and request is being filed via EFS-Web.
iii. The application is an original nonprovisional utility application filed under 35 U.S.C. 111(a), or is a national stage entry under 35 U.S.C. 371.
iv. This certification and request is being filed prior to the mailing of a first Office action responsive to the request for continued examination.
v. No prior request for continued examination has been granted prioritized examination status under 37 CFR 1.102(e)(2).

Signature /BRETT A. CARLSON/	Date 2020-05-18
Name (Print/Typed) Brett A. Carlson	Practitioner Registration Number 39,928

Note: *Signatures of all the inventors or assignees of record of the entire interest or their representative(s) are required in accordance with 37 CFR 1.33 and 11.18. Please see 37 CFR 1.4(d) for the form of the signature. If necessary, submit multiple forms for more than one signature, see below*.*

☑ *Total of 1 forms are submitted.

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.

Doc Code: PA..

Document Description: Power of Attorney

PTO/AIA/82A (07-12)

Approved for use through 11/30/2014. OMB 0651-0035

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 displays a valid OMB control number.

TRANSMITTAL FOR POWER OF ATTORNEY TO ONE OR MORE REGISTERED PRACTITIONERS

NOTE: This form is to be submitted with the Power of Attorney by Applicant form (PTO/AIA/82B or equivalent) to identify the application to which the Power of Attorney is directed, in accordance with 37 CFR 1.5. If the Power of Attorney by Applicant form is not accompanied by this transmittal form or an equivalent, the Power of Attorney will not be recognized in the application.

Application Number	Unassigned
Filing Date	Unassigned
First Named Inventor	DAVID F. BRUECK
Title	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING
Art Unit	Unassigned
Examiner Name	
Attorney Docket Number	066.0141X1C9

SIGNATURE of Applicant or Patent Practitioner

Signature	/BRETT A. CARLSON/	Date	2020-05-18
Name	Brett A. Carlson	Telephone	480.385.5060
Registration Number	39,928		

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

[X] *Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.31, 1.32 and 1.33. 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 1.11 and 1.14. This collection is estimated to take 3 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, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, 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.

Doc Code: PA.
Document Description: Power of Attorney

PTO/AIA/82B (07-13)
Approved for use through 11/30/2014. 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 displays a valid OMB control number

POWER OF ATTORNEY BY APPLICANT

I hereby revoke all previous powers of attorney given in the application identified in either the attached transmittal letter or the boxes below.

Application Number	Filing Date

(Note: The boxes above may be left blank if information is provided on form PTO/AIA/82A.)

☑ I hereby appoint the Patent Practitioner(s) associated with the following Customer Number as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above: 70560

OR

☐ I hereby appoint Practitioner(s) named in the attached list (form PTO/AIA/82C) as my/our attorney(s) or agent(s), and to transact all business in the United States Patent and Trademark Office connected therewith for the patent application referenced in the attached transmittal letter (form PTO/AIA/82A) or identified above. (Note: Complete form PTO/AIA/82C.)

Please recognize or change the correspondence address for the application identified in the attached transmittal letter or the boxes above to:

☑ The address associated with the above-mentioned Customer Number

OR

☐ The address associated with Customer Number:

OR

Firm or Individual Name					
Address					
City		State		Zip	
Country					
Telephone		Email			

I am the Applicant (if the Applicant is a juristic entity, list the Applicant name in the box):

DISH Technologies L.L.C.

☐ Inventor or Joint Inventor (title not required below)

☐ Legal Representative of a Deceased or Legally Incapacitated Inventor (title not required below)

☑ Assignee or Person to Whom the Inventor is Under an Obligation to Assign (provide signer's title if applicant is a juristic entity)

☐ Person Who Otherwise Shows Sufficient Proprietary Interest (e.g., a petition under 37 CFR 1.46(b)(2) was granted in the application or is concurrently being filed with this document) (provide signer's title if applicant is a juristic entity)

SIGNATURE of Applicant for Patent

The undersigned (whose title is supplied below) is authorized to act on behalf of the applicant (e.g., where the applicant is a juristic entity).

Signature	/Max Gratton/	Date (Optional)	7/6/2016
Name	Max S. Gratton		
Title	Director & Senior Corporate Counsel of DISH Technologies L.L.C.		

NOTE: Signature - This form must be signed by the applicant in accordance with 37 CFR 1.33. See 37 CFR 1.4 for signature requirements and certifications. If more than one applicant, use multiple forms.

☑ Total of 1 forms are submitted.

This collection of information is required by 37 CFR 1.131, 1.32, and 1.33. 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 1.11 and 1.14. This collection is estimated to take 3 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, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. SEND TO: Commissioner for Patents, 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.

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Serial. No.:	TBD	Confirmation No.:	TBD
Examiner:	TBD	Art Unit:	TBD
Filed:	Herewith	Customer No.:	70560

Title: **APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING**

Attorney Docket No.: 066.0141x1C9

PRELIMINARY AMENDMENT

Mail Stop AMENDMENT
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

****FILED ELECTRONICALLY USING THE PTO-EFS SYSTEM***

Commissioner:

Please amend the above-referenced application prior to examination.

Amendments to the Specification begin on the following page (page 2).

Remarks begin on page 3 of this paper.

REMARKS

This preliminary amendment is being filed at the same time as the attached application to update the priority information in paragraph 0001. Please enter the foregoing amendments prior to substantive examination. No new matter has been added, and support for all of the claimed features can be found in the originally-filed application.

This application is being filed as a "Track One" application with a certification and request for prioritized examination. If any documents or fees that are required to process this application as a Track One application are not submitted herewith, ***please consider this as a petition under Rule 181 (or any other appropriate rule) for appropriate relief, and as authorization to charge Deposit Account No. 50-2091 for any fees that may be due***, including any fees to process the Track One application and any associated petitions or other documents.

Should the Examiner have any suggestions that would advance prosecution at any time during prosecution, we would welcome a telephone call to (480) 385-5069 or an email to bcarlson@lkglobal.com. *Recognizing that Internet communications are not secure, we hereby authorize the USPTO to communicate with the undersigned and practitioners in accordance with 37 CFR 1.33 and 37 CFR 1.34 concerning any subject matter of this application via video conferencing, instant messaging, or electronic mail. We understand that a copy of these communications will be made of record in the application file. (MPEP 502.03). Without limiting the foregoing, we further authorize the use of email for informal correspondence, including the exchange of proposed amendments or other draft documents.*

If any additional documents or fees are required to consider this submittal or to otherwise prevent abandonment of this application, please consider this as a petition and request for any needed relief and as authorization to charge Deposit Account No. 50-2091 for any fees that may be due.

Respectfully submitted,

LORENZ & KOPF, LLC

Dated: May 18, 2020

By: /Brett A. Carlson /
Brett A. Carlson
Reg. No. 39,928

bcarlson@lkglobal.com
480-385-5069

LORENZ & KOPF, LLC (*formerly Ingrassia, Fisher & Lorenz, PC*)
Customer No. 70560

AMENDMENTS TO THE SPECIFICATION

PLEASE AMEND PARAGRAPH 0001 OF THE SPECIFICATION AS FOLLOWS:

[0001] This application is a continuation of U.S. Patent Application Serial No. 16/004,056 filed on June 8, 2018, which is a continuation of U.S. Patent Application Serial No. 15/414,025 (now US Patent No. 9,998,516) filed on January 24, 2017, which is a continuation of U.S. Patent Application No. 14/719,122 filed on May 21, 2015, which is a continuation of U.S. Patent Application No. 14/106,051 filed on December 13, 2013 (now U.S. Patent No. 9,071,668), which is a continuation of U.S. Patent Application Serial No. 13/617,114, filed on September 14, 2012 (now US Patent No. 8,612,624), which is a continuation of U.S. Patent No. 12/906,940 filed on October 18, 2010 (now US Patent No. 8,402,156), which is a continuation of U.S. Patent Application Number 11/673,483, filed on February 9, 2007 (now US Patent No. 7,818,444), which is a continuation-in-part of Application No. 11/116,783, filed on April 28, 2005 (now U.S. Patent No. 8,868,772), which claims the benefit of U.S. Provisional Application No. 60/566,831, filed on April 31, 2004, all of which are incorporated herein by reference.

Electronic Acknowledgement Receipt

EFS ID:	39461774
Application Number:	16876579
International Application Number:	
Confirmation Number:	3720
Title of Invention:	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING
First Named Inventor/Applicant Name:	DAVID F. BRUECK
Customer Number:	70560
Filer:	Brett Carlson/Julie Graf
Filer Authorized By:	Brett Carlson
Attorney Docket Number:	066.0141X1C9
Receipt Date:	18-MAY-2020
Filing Date:	
Time Stamp:	14:57:41
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	CARD
Payment was successfully received in RAM	$7320
RAM confirmation Number	E20205HE58013692
Deposit Account	
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Application Data Sheet	20200518_ADS_066-0141X1C9.pdf	1823905 ef14709de8eafcfe97ceb6d9649c489ab6cf3b17	no	10
Warnings:					
Information:					
2		20200518_ConAppl_066-0141X1C9_P2007-02-01-2-7.pdf	14773111 5a0b50a0c154548893bd585064128b616e07fb98	yes	36

	Multipart Description/PDF files in .zip description		
	Document Description	**Start**	**End**
	Specification	1	28
	Claims	29	35
	Abstract	36	36

Warnings:					
Information:					
3	Oath or Declaration filed	20200518_DeclarationOfJointInvs_066-0141X1C9.pdf	135215 98d341969b7e66dbccf545b2f138fadf294d0db5	no	3
Warnings:					
Information:					
4	Drawings-only black and white line drawings	20200518_Dwgs_066-0141X1C9.pdf	133801 e3ec7077c08c48b61e561c9ae9f1f2b97f7a3624	no	11
Warnings:					
Information:					
5	Power of Attorney	20200518_POA_066-0141X1C9.pdf	114418 3432aef93cb44197661de60bce07113ab6c3a0aa	no	2

Warnings:					
The page size in the PDF is too large. The pages should be 8.5 x 11 or A4. If this PDF is submitted, the pages will be resized upon entry into the Image File Wrapper and may affect subsequent processing					
Information:					
6	TrackOne Request	20200518_ReqForExam.pdf	140703 4fe05ab6a296bdbdcd3c961540843bae0365445c	no	2
Warnings:					
Information:					
7	Preliminary Amendment	20200518_PreliminaryAmendment_06-0141X1C9.pdf	149179 c9ade1c8adee2b3a97df2a4b7bb94ef85ca4c296	no	4
Warnings:					
Information:					
8	Fee Worksheet (SB06)	fee-info.pdf	43027 5097de90893fd873f8e9fac2314314f1bad06efe	no	2
Warnings:					
Information:					
		Total Files Size (in bytes):	17313359		

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.

APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING

CROSS-REFERENCES TO RELATED APPLICATIONS

[0001] This application is a continuation of U.S. Patent Application Number 11/673,483, filed on February 9, 2007, which is a continuation-in-part of Application No. 11/116,783, filed on April 28, 2005, which claims the benefit of U.S. Provisional Application No. 60/566,831, filed on April 30, 2004, all of which are incorporated herein by reference.

BACKGROUND OF THE INVENTION

FIELD OF THE INVENTION

[0002] The invention relates to video streaming over packet switched networks such as the Internet, and more particularly relates to adaptive-rate shifting of streaming content over such networks.

DESCRIPTION OF THE RELATED ART

[0003] The Internet is fast becoming a preferred method for distributing media files to end users. It is currently possible to download music or video to computers, cell phones, or practically any network capable device. Many portable media players are equipped with network connections and enabled to play music or videos. The music or video files (hereinafter "media files") can be stored locally on the media player or computer, or streamed or downloaded from a server.

[0004] "Streaming media" refers to technology that delivers content at a rate sufficient for presenting the media to a user in real time as the data is received. The data may be stored in memory temporarily until played and then subsequently deleted. The user has the immediate satisfaction of viewing the requested content without waiting for the media file to completely download. Unfortunately, the audio/video quality that can be received for real time presentation is constrained by the available bandwidth of the user's network connection. Streaming may be used to deliver content on demand (previously recorded) or from live broadcasts.

[0005] Alternatively, media files may be downloaded and stored on persistent storage devices, such as hard drives or optical storage, for later presentation. Downloading complete media files can take large amounts of time depending on the network connection. Once downloaded, however, the content can be viewed repeatedly anytime or anywhere. Media files prepared for downloading usually are encoded with a higher quality audio/video than can be delivered in real time. Users generally dislike this option, as they tend to want to see or hear the media file instantaneously.

[0006] Streaming offers the advantage of immediate access to the content but currently sacrifices quality compared with downloading a file of the same content. Streaming also provides the opportunity for a user to select different content for viewing on an ad hoc basis, while downloading is by definition restricted to receiving a specific content selection in its entirety or not at all. Downloading also supports rewind, fast forward, and direct seek operations, while streaming is unable to fully support these functions. Streaming is also vulnerable to network failures or congestion.

[0007] Another technology, known as "progressive downloads," attempts to combine the strengths of the above two technologies. When a progressive download is initiated, the media file download begins, and the media player waits to begin playback until there is enough of the file downloaded that playback can begin with the hope that the remainder of the file will be completely downloaded before playback "catches up." This waiting period before playback can be substantial depending on network conditions, and therefore is not a complete or fully acceptable solution to the problem of media presentation over a network.

[0008] Generally, three basic challenges exist with regard to data transport streaming over a network such as the Internet that has a varying amount of data loss. The first challenge is reliability. Most streaming solutions use a TCP connection, or "virtual circuit," for transmitting data. A TCP connection provides a guaranteed delivery mechanism so that data sent from one endpoint will be delivered to the destination, even if portions are lost and retransmitted. A break in the continuity of a TCP connection can have serious consequences when the data must be delivered in real-time. When a network adapter detects delays or losses in a TCP connection, the adapter "backs off" from transmission attempts for a moment and then slowly resumes the original

transmission pace. This behavior is an attempt to alleviate the perceived congestion. Such a slowdown is detrimental to the viewing or listening experience of the user and therefore is not acceptable.

[0009] The second challenge to data transport is efficiency. Efficiency refers to how well the user's available bandwidth is used for delivery of the content stream. This measure is directly related to the reliability of the TCP connection. When the TCP connection is suffering reliability problems, a loss of bandwidth utilization results. The measure of efficiency sometimes varies suddenly, and can greatly impact the viewing experience.

[0010] The third challenge is latency. Latency is the time measure form the client's point-of-view, of the interval between when a request is issued and the response data begins to arrive. This value is affected by the network connection's reliability and efficiency, and the processing time required by the origin to prepare the response. A busy or overloaded server, for example, will take more time to process a request. As well as affecting the start time of a particular request, latency has a significant impact on the network throughput of TCP.

[0011] From the foregoing discussion, it should be apparent that a need exists for an apparatus, system, and method that alleviate the problems of reliability, efficiency, and latency. Additionally, such an apparatus, system, and method would offer instantaneous viewing along with the ability to fast forward, rewind, direct seek, and browse multiple streams. Beneficially, such an apparatus, system, and method would utilize multiple connections between a source and destination, requesting varying bitrate streams depending upon network conditions.

SUMMARY OF THE INVENTION

[0012] The present invention has been developed in response to the present state of the art, and in particular, in response to the problems and needs in the art that have not yet been fully solved by currently available content streaming systems. Accordingly, the present invention has been developed to provide an apparatus, system, and method for adaptive-rate content streaming that overcome many or all of the above-discussed shortcomings in the art.

[0013] The apparatus for adaptive-rate content streaming is provided with a logic unit containing a plurality of modules configured to functionally execute the necessary steps. These modules in the described embodiments include a receiving module configured to receive media content, a streamlet module configured to segment the media content and generate a plurality of sequential streamlets, and an encoding module configured to encode each streamlet as a separate content file.

[0014] The encoding module is further configured to generate a set of streamlets for each of the sequential streamlets. Each streamlet may comprise a portion of the media content having a predetermined length of time. The predetermined length of time may be in the range of between about .1 and 5 seconds.

[0015] In one embodiment, a set of streamlets comprises a plurality of streamlets having identical time indices, and each streamlet of the set of streamlets has a unique bitrate. The receiving module is configured to convert the media content to raw audio or raw video. The encoding module may include a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid. The job completion bid may be based on a plurality of computing variables selected from a group consisting of current encoding job completion percentage, average encoding job completion time, processor speed, and physical memory capacity.

[0016] A system of the present invention is also presented for adaptive-rate content streaming. In particular, the system, in one embodiment, includes a receiving module configured to receive media content, a streamlet module configured to segment the media content and generate a plurality of sequential streamlets, each streamlet comprising a portion of the media content having a predetermined length of time, and an encoding

module configured to encode each streamlet as a separate content file and generate a set of streamlets.

[0017] The system also includes a plurality of streamlets having identical time indices and each streamlet of the set of streamlets having a unique bitrate. The encoding module comprises a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid.

[0018] A method of the present invention is also presented for adaptive-rate content streaming. In one embodiment, the method includes receiving media content, segmenting the media content and generating a plurality of sequential streamlets, and encoding each streamlet as a separate content file.

[0019] The method also includes segmenting the media content into a plurality of streamlets, each streamlet comprising a portion of the media content having a predetermined length of time. In one embodiment, the method includes generating a set of streamlets comprising a plurality of streamlets having identical time indices, and each streamlet of the set of streamlets having a unique bitrate.

[0020] Furthermore, the method may include converting the media content to raw audio or raw video, and segmenting the content media into a plurality of sequential streamlets. The method further comprises assigning an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid, and submitting an encoding job completion bid based on a plurality of computing variables.

[0021] Reference throughout this specification to features, advantages, or similar language does not imply that all of the features and advantages that may be realized with the present invention should be or are in any single embodiment of the invention. Rather, language referring to the features and advantages is understood to mean that a specific feature, advantage, or characteristic described in connection with an embodiment is included in at least one embodiment of the present invention. Thus, discussion of the features and advantages, and similar language, throughout this specification may, but do not necessarily, refer to the same embodiment.

[0022] Furthermore, the described features, advantages, and characteristics of the invention may be combined in any suitable manner in one or more embodiments. One skilled in the relevant art will recognize that the invention may be practiced without one

or more of the specific features or advantages of a particular embodiment. In other instances, additional features and advantages may be recognized in certain embodiments that may not be present in all embodiments of the invention.

[0023] These features and advantages of the present invention will become more fully apparent from the following description and appended claims, or may be learned by the practice of the invention as set forth hereinafter.

BRIEF DESCRIPTION OF THE DRAWINGS

[0024] In order that the advantages of the invention will be readily understood, a more particular description of the invention briefly described above will be rendered by reference to specific embodiments that are illustrated in the appended drawings. Understanding that these drawings depict only typical embodiments of the invention and are not therefore to be considered to be limiting of its scope, the invention will be described and explained with additional specificity and detail through the use of the accompanying drawings, in which:

[0025] Figure 1 is a schematic block diagram illustrating one embodiment of a system for dynamic rate shifting of streaming content in accordance with the present invention;

[0026] Figure 2a is a schematic block diagram graphically illustrating one embodiment of a media content file;

[0027] Figure 2b is a schematic block diagram illustrating one embodiment of a plurality of streams having varying degrees of quality and bandwidth;

[0028] Figure 3a is a schematic block diagram illustrating one embodiment of a stream divided into a plurality of source streamlets;

[0029] Figure 3b is a schematic block diagram illustrating one embodiment of sets of streamlets in accordance with the present invention;

[0030] Figure 4 is a schematic block diagram illustrating in greater detail one embodiment of the content module in accordance with the present invention;

[0031] Figure 5a is a schematic block diagram illustrating one embodiment of an encoder module in accordance with the present invention;

[0032] Figure 5b is a schematic block diagram illustrating one embodiment of parallel encoding of streamlets in accordance with the present invention;

[0033] Figure 6a is a schematic block diagram illustrating one embodiment of a virtual timeline in accordance with the present invention;

[0034] Figure 6b is a schematic block diagram illustrating an alternative embodiment of a VT in accordance with the present invention;

[0035] Figure 6c is a schematic block diagram illustrating one embodiment of a QMX in accordance with the present invention;

[0036] Figure 7 is a schematic block diagram graphically illustrating one embodiment of a client module in accordance with the present invention;

[0037] Figure 8 is a schematic flow chart diagram illustrating one embodiment of a method for processing content in accordance with the present invention;

[0038] Figure 9 is a schematic flow chart diagram illustrating one embodiment of a method for viewing a plurality of streamlets in accordance with the present invention; and

[0039] Figure 10 is a schematic flow chart diagram illustrating one embodiment of a method for requesting streamlets within an adaptive-rate shifting content streaming environment in accordance with the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0040] Many of the functional units described in this specification have been labeled as modules, in order to more particularly emphasize their implementation independence. For example, a module may be implemented as a hardware circuit comprising custom VLSI circuits or gate arrays, off-the-shelf semiconductors such as logic chips, transistors, or other discrete components. A module may also be implemented in programmable hardware devices such as field programmable gate arrays, programmable array logic, programmable logic devices or the like.

[0041] Modules may also be implemented in software for execution by various types of processors. An identified module of executable code may, for instance, comprise one or more physical or logical blocks of computer instructions which may, for instance, be organized as an object, procedure, or function. Nevertheless, the executables of an identified module need not be physically located together, but may comprise disparate instructions stored in different locations which, when joined logically together, comprise the module and achieve the stated purpose for the module.

[0042] Indeed, a module of executable code may be a single instruction, or many instructions, and may even be distributed over several different code segments, among different programs, and across several memory devices. Similarly, operational data may be identified and illustrated herein within modules, and may be embodied in any suitable form and organized within any suitable type of data structure. The operational data may be collected as a single data set, or may be distributed over different locations including over different storage devices, and may exist, at least partially, merely as electronic signals on a system or network.

[0043] Reference throughout this specification to "one embodiment," "an embodiment," or similar language means that a particular feature, structure, or characteristic described in connection with the embodiment is included in at least one embodiment of the present invention. Thus, appearances of the phrases "in one embodiment," "in an embodiment," and similar language throughout this specification may, but do not necessarily, all refer to the same embodiment.

[0044] Reference to a signal bearing medium may take any form capable of generating a signal, causing a signal to be generated, or causing execution of a program of machine-

readable instructions on a digital processing apparatus. A signal bearing medium may be embodied by a transmission line, a compact disk, digital-video disk, a magnetic tape, a Bernoulli drive, a magnetic disk, a punch card, flash memory, integrated circuits, or other digital processing apparatus memory device. In one embodiment, a computer program product including a computer useable medium having a computer readable program of computer instructions stored thereon that when executed on a computer causes the computer to carry out operations for multi-bitrate content streaming as described herein.

[0045] Furthermore, the described features, structures, or characteristics of the invention may be combined in any suitable manner in one or more embodiments. In the following description, numerous specific details are provided, such as examples of programming, software modules, user selections, network transactions, database queries, database structures, hardware modules, hardware circuits, hardware chips, etc., to provide a thorough understanding of embodiments of the invention. One skilled in the relevant art will recognize, however, that the invention may be practiced without one or more of the specific details, or with other methods, components, materials, and so forth. In other instances, well-known structures, materials, or operations are not shown or described in detail to avoid obscuring aspects of the invention.

[0046] Figure 1 is a schematic block diagram illustrating one embodiment of a system 100 for dynamic rate shifting of streaming content in accordance with the present invention. In one embodiment, the system 100 comprises a content server 102 and an end user station 104. The content server 102 and the end user station 104 may be coupled by a data communications network. The data communications network may include the Internet 106 and connections 108 to the Internet 106. Alternatively, the content server 102 and the end user 104 may be located on a common local area network, wireless area network, cellular network, virtual local area network, or the like. The end user station 104 may comprise a personal computer (PC), an entertainment system configured to communicate over a network, or a portable electronic device configured to present content. For example, portable electronic devices may include, but are not limited to, cellular phones, portable gaming systems, and portable computing devices.

[0047] In the depicted embodiment, the system 100 also includes a publisher 110, and a web server 116. The publisher 110 may be a creator or distributor of content. For

example, if the content to be streamed were a broadcast of a television program, the publisher 110 may be a television or cable network channel such as NBC®, or MTV®. Content may be transferred over the Internet 106 to the content server 102, where the content is received by a content module 112. The content module 112 may be configured to receive, process, and store content. In one embodiment, processed content is accessed by a client module 114 configured to play the content on the end user station 104. In a further embodiment, the client module 114 is configured to receive different portions of a content stream from a plurality of locations simultaneously. For example, the client module 114 may request and receive content from any of the plurality of web servers 116.

[0048] Content from the content server 102 may be replicated to other web servers 116 or alternatively to proxy cache servers 118. Replicating may occur by deliberate forwarding from the content server 102, or by a web, cache, or proxy server outside of the content server 102 asking for content on behalf of the client module 114. In a further embodiment, content may be forwarded directly to web 116 or proxy 118 servers through direct communication channels 120 without the need to traverse the Internet 106.

[0049] Figure 2a is a schematic block diagram graphically illustrating one embodiment of a media content (hereinafter "content") file 200. In one embodiment, the content file 200 is distributed by the publisher 110. The content file 200 may comprise a television broadcast, sports event, movie, music, concert, etc. The content file 200 may also be live or archived content. The content file 200 may comprise uncompressed video and audio, or alternatively, video or audio. Alternatively, the content file 200 may be compressed using standard or proprietary encoding schemes. Examples of encoding schemes capable of use with the present invention include, but are not limited to, DivX®, Windows Media Video®, Quicktime Sorenson 3®, On2, OGG Vorbis, MP3, or Quicktime 6.5/ MPEG-4® encoded content.

[0050] Figure 2b is a schematic block diagram illustrating one embodiment of a plurality of streams 202 having varying degrees of quality and bandwidth. In one embodiment, the plurality of streams 202 comprises a low quality stream 204, a medium quality stream 206, and a high quality stream 208. Each of the streams 204, 206, 208 is a copy of the content file 200 encoded and compressed to varying bit rates. For example, the low quality stream 204 may be encoded and compressed to a bit rate of 100 kilobits per

second (kbps), the medium quality stream 206 may be encoded and compressed to a bit rate of 200 kbps, and the high quality stream 208 may be encoded and compressed to 600 kbps.

[0051] Figure 3a is a schematic block diagram illustrating one embodiment of a stream 302 divided into a plurality of source streamlets 303. As used herein, streamlet refers to any sized portion of the content file 200. Each streamlet 303 may comprise a portion of the content contained in stream 302, encapsulated as an independent media object. The content in a streamlet 303 may have a unique time index in relation to the beginning of the content contained in stream 302. In one embodiment, the content contained in each streamlet 303 may have a duration of two seconds. For example, streamlet 0 may have a time index of 00:00 representing the beginning of content playback, and streamlet 1 may have a time index of 00:02, and so on. Alternatively, the time duration of the streamlets 304 may be any duration smaller than the entire playback duration of the content in stream 302. In a further embodiment, the streamlets 303 may be divided according to file size instead of a time index and duration.

[0052] Figure 3b is a schematic block diagram illustrating one embodiment of sets 306 of streamlets in accordance with the present invention. As used herein, the term "set" refers to a group of streamlets having identical time indices and durations but varying bitrates. In the depicted embodiment, the set 306a encompasses all streamlets having a time index of 00:00. The set 306a includes encoded streamlets 304 having low, medium, and high 204, 206, 208 bitrates. Of course each set 306 may include more than the depicted three bitrates which are given by way of example only. One skilled in the art will recognize that any number of streams having different bitrates may be generated from the original content 200.

[0053] As described above, the duration of one streamlet 304 may be approximately two seconds. Likewise each set 306 may comprise a plurality of streamlets 304 where each streamlet 304 has a playable duration of two seconds. Alternatively, the duration of the streamlet 304 may be predetermined or dynamically variable depending upon a variety of factors including, but not limited to, network congestion, system specifications, playback resolution and quality, etc. In the depicted embodiment, the content 200 may be formed

of the plurality of sets 306. The number of sets 306 may depend on the length of the content 200 and the length or duration of each streamlet 304.

[0054] Figure 4 is a schematic block diagram illustrating in greater detail one embodiment of the content module 112 in accordance with the present invention. The content module 112 may comprise a capture module 402, a streamlet module 404, an encoder module 406, a streamlet database 408, and the web server 116. In one embodiment, the capture module 402 is configured to receive the content file 200 from the publisher 110. The capture module 402 may be configured to "decompress" the content file 200. For example, if the content file 200 arrives having been encoded with one of the above described encoding schemes, the capture module 402 may convert the content file 200 into raw audio and/or video. Alternatively, the content file 200 may be transmitted by the publisher in a format 110 that does not require decompression.

[0055] The capture module 402 may comprise a capture card configured for TV and/or video capture. One example of a capture card suitable for use in the present invention is the DRC-2500 by Digital Rapids of Ontario, Canada. Alternatively, any capture card capable of capturing audio and video may be utilized with the present invention. In a further embodiment, the capture module 402 is configured to pass the content file to the streamlet module 404.

[0056] The streamlet module 404, in one embodiment, is configured to segment the content file 200 and generate source streamlets 303 that are not encoded. As used herein, the term "segment" refers to an operation to generate a streamlet of the content file 200 having a duration or size equal to or less than the duration or size of the content file 200. The streamlet module 404 may be configured to segment the content file 200 into streamlets 303 each having an equal duration. Alternatively, the streamlet module 404 may be configured to segment the content file 200 into streamlets 303 having equal file sizes.

[0057] The encoding module 406 is configured to receive the source streamlets 303 and generate the plurality of streams 202 of varying qualities. The original content file 200 from the publisher may be digital in form and may comprise content having a high bit rate such as, for example, 2mbps. The content may be transferred from the publisher 110 to the content module 112 over the Internet 106. Such transfers of data are well known in

the art and do not require further discussion herein. Alternatively, the content may comprise a captured broadcast.

[0058] In a further embodiment, the encoding module 406 is configured to generate a plurality of sets 306 of streamlets 304. The sets 306, as described above with reference to Figure 3b, may comprise streamlets having an identical time index and duration, and a unique bitrate. As with Figure 3b, the sets 306 and subsequently the plurality of streams 202 may comprise the low quality stream 204, the medium quality stream 206, and the high quality stream 208. Alternatively, the plurality of streams 202 may comprise any number of streams deemed necessary to accommodate end user bandwidth.

[0059] The encoder module 406 is further configured to encode each source streamlet 303 into the plurality of streams 202 and streamlet sets 306 and store the streamlets in the streamlet database 408. The encoding module 406 may utilize encoding schemes such as DivX®, Windows Media Video 9®, Quicktime 6.5 Sorenson 3®, or Quicktime 6.5/ MPEG-4®. Alternatively, a custom encoding scheme may be employed.

[0060] The content module 112 may also include a metadata module 412 and a metadata database 414. In one embodiment, metadata comprises static searchable content information. For example, metadata includes, but is not limited to, air date of the content, title, actresses, actors, length, and episode name. Metadata is generated by the publisher 110, and may be configured to define an end user environment. In one embodiment, the publisher 100 may define an end user navigational environment for the content including menus, thumbnails, sidebars, advertising, etc. Additionally, the publisher 110 may define functions such as fast forward, rewind, pause, and play that may be used with the content file 200. The metadata module 412 is configured to receive the metadata from the publisher 110 and store the metadata in the metadata database 414. In a further embodiment, the metadata module 412 is configured to interface with the client module 114, allowing the client module 114 to search for content based upon at least one of a plurality of metadata criteria. Additionally, metadata may be generated by the content module 112 through automated process(es) or manual definition.

[0061] Once the streamlets 304 have been received and processed, the client module 114 may request streamlets 304 using HTTP from the web server 116. Using a standard protocol such as HTTP eliminates the need for network administrators to configure

firewalls to recognize and pass through network traffic for a new, specialized protocol. Additionally, since the client module 114 initiates the request, the web server 116 is only required to retrieve and serve the requested streamlet 304. In a further embodiment, the client module 114 may be configured to retrieve streamlets 304 from a plurality of web servers 116.

[0062] Each web server 116 may be located in various locations across the Internet 106. The streamlets 304 may essentially be static files. As such, no specialized media server or server-side intelligence is required for a client module 114 to retrieve streamlets 304. Streamlets 304 may be served by the web server 116 or cached by cache servers of Internet Service Providers (ISPs), or any other network infrastructure operators, and served by the cache server. Use of cache servers is well known to those skilled in the art, and will not be discussed further herein. Thus, a highly scalable solution is provided that is not hindered by massive amounts of client module 114 requests to the web server 116 at any specific location, especially the web server 116 most closely associated with or within the content module 112

[0063] Figure 5a is a schematic block diagram illustrating one embodiment of an encoder module 406 in accordance with the present invention. In one embodiment, the encoder module 406 may include a master module 502 and a plurality of host computing modules (hereinafter "host") 504. The hosts 504 may comprise personal computers, servers, etc. In a further embodiment, the hosts 504 may be dedicated hardware, for example, cards plugged into a single computer.

[0064] The master module (hereinafter "master") 502 is configured to receive streamlets 303 from the streamlet module 404 and stage the streamlet 303 for processing. In one embodiment, the master 502 may decompress each source streamlet 303 to produce a raw streamlet. As used herein, the term "raw streamlet" refers to a streamlet 303 that is uncompressed or lightly compressed to substantially reduce size with no significant loss in quality. A lightly compressed raw streamlet can be transmitted more quickly and to more hosts. Each host 504 is coupled with the master 502 and configured to receive a raw streamlet from the master 502 for encoding. The hosts 504, in one example, generate a plurality of streamlets 304 having identical time indices and durations, and varying bitrates. Essentially each host 504 may be configured to generate a set 306 from the raw

streamlet 503 sent from the master 502. Alternatively, each host 504 may be dedicated to producing a single bitrate in order to reduce the time required for encoding.

[0065] Upon encoding completion, the host 504 returns the set 306 to the master 502 so that the encoding module 406 may store the set 306 in the streamlet database 408. The master 502 is further configured to assign encoding jobs to the hosts 504. Each host is configured to submit an encoding job completion bid (hereinafter "bid"). The master 502 assigns encoding jobs depending on the bids from the hosts 504. Each host 504 generates a bid depending upon a plurality of computing variables which may include, but are not limited to, current encoding job completion percentage, average job completion time, processor speed and physical memory capacity.

[0066] For example, a host 504 may submit a bid that indicates that based on past performance history the host 504 would be able to complete the encoding job in 15 seconds. The master 502 is configured to select from among a plurality of bids the best bid and subsequently submit the encoding job to the host 504 with the best bid. As such, the described encoding system does not require that each host 504 have identical hardware but beneficially takes advantage of the available computing power of the hosts 504. Alternatively, the master 502 selects the host 504 based on a first come first serve basis, or some other algorithm deemed suitable for a particular encoding job.

[0067] The time required to encode one streamlet 304 is dependent upon the computing power of the host 504, and the encoding requirements of the content file 200. Examples of encoding requirements may include, but are not limited to, two or multi-pass encoding, and multiple streams of different bitrates. One benefit of the present invention is the ability to perform two-pass encoding on a live content file 200. Typically, in order to perform two-pass encoding prior art systems must wait for the content file to be completed before encoding.

[0068] The present invention, however, segments the content file 200 into source streamlets 303 and the two-pass encoding to a plurality of streams 202 may be performed on each corresponding raw streamlet without waiting for a TV show to end, for example. As such, the content module 112 is capable of streaming the streamlets over the Internet shortly after the content module 112 begins capture of the content file 200. The delay

between a live broadcast transmitted from the publisher 110 and the availability of the content depends on the computing power of the hosts 504.

[0069] Figure 5b is a schematic block diagram illustrating one embodiment of parallel encoding of streamlets in accordance with the present invention. In one example, the capture module 402 (of Figure 4) begins to capture the content file and the streamlet module 404 generates a first streamlet 303a and passes the streamlet to the encoding module 406. The encoding module 406 may take 10 seconds, for example, to generate the first set 306a of streamlets 304a (304a1, 304a2, 304a3, etc. represent streamlets 304 of different bitrates). Figure 5b illustrates the encoding process generically as block 502 to graphically illustrate the time duration required to process a raw or lightly encoded streamlet 303 as described above with reference to the encoding module 406. The encoding module 406 may simultaneously process more than one streamlet 303, and processing of streamlets will begin upon arrival of the streamlet from the capture module 402.

[0070] During the 10 seconds required to encode the first streamlet 303a, the streamlet module 404 has generated five additional 2-second streamlets 303b, 303c, 303d, 303e, 303f, for encoding and the master 502 has prepared and staged the corresponding raw streamlets. Two seconds after the first set 306a is available the next set 306b is available, and so on. As such, the content file 200 is encoded for streaming over the Internet and appears live. The 10 second delay is given herein by way of example only. Multiple hosts 504 may be added to the encoding module 406 in order to increase the processing capacity of the encoding module 406. The delay may be shortened to an almost unperceivable level by the addition of high CPU powered systems, or alternatively multiple low powered systems.

[0071] A system as described above beneficially enables multi-pass encoding of live events. Multi-pass encoding systems of the prior art require that the entire content be captured (or be complete) because in order to perform multi-pass encoding the entire content must be scanned and processed more than once. This is impossible with prior art systems because content from a live event is not complete until the event is over. As such, with prior art systems, multi-pass encoding can only be performed once the event is over. Streamlets, however, may be encoded as many times as is deemed necessary.

Because the streamlet is an encapsulated media object of 2 seconds (for example), multi-pass encoding may begin on a live event once the first streamlet is captured. Shortly after multi-pass encoding of the first streamlet 303a is finished, multi-pass encoding of the second streamlet 303b finishes, and as such multi-pass encoding is performed on a live event and appears live to a viewer.

[0072] Any specific encoding scheme applied to a streamlet may take longer to complete than the time duration of the streamlet itself, for example, a very high quality encoding of a 2-second streamlet may take 5 seconds to finish. Alternatively, the processing time required for each streamlet may be less than the time duration of a streamlet. However, because the offset parallel encoding of successive streamlets are encoded by the encoding module at regular intervals (matching the intervals at which the those streamlets are submitted to the encoding module 406, for example 2 seconds) the output timing of the encoding module 406 does not fall behind the real-time submission rate of the unencoded streamlets. Conversely, prior art encoding systems rely on the very fastest computing hardware and software because the systems must generate the output immediately in lock-step with the input. A prior art system that takes 2.1 seconds to encode 2 seconds worth of content is considered a failure. The present invention allows for slower than real-time encoding processes yet still achieves a real-time encoding effect due to the parallel offset pipes.

[0073] The parallel offset pipeline approach described with reference to Figure 5b beneficially allows for long or short encoding times without "falling behind" the live event. Additionally, arbitrarily complex encoding of streamlets to multiple profiles and optimizations only lengthens the encoding time 502 without a perceptible difference to a user because the sets 306 of streamlets 304 are encoded in a time-selective manner so that streamlets are processed at regular time intervals and transmitted at these time intervals.

[0074] Returning now to Figure 5a, as depicted, the master 502 and the hosts 504 may be located within a single local area network, or in other terms, the hosts 504 may be in close physical proximity to the master 502. Alternatively, the hosts 504 may receive encoding jobs from the master 502 over the Internet or other communications network. For example, consider a live sports event in a remote location where it would be difficult to setup multiple hosts. In this example, a master performs no encoding or alternatively

light encoding before publishing the streamlets online. The hosts 504 would then retrieve those streamlets and encode the streamlets into the multiple bitrate sets 306 as described above.

[0075] Furthermore, hosts 504 may be dynamically added or removed from the encoding module without restarting the encoding job and/or interrupting the publishing of streamlets. If a host 504 experiences a crash or some failure, its encoding work is simply reassigned to another host.

[0076] The encoding module 406, in one embodiment, may also be configured to produce streamlets that are specific to a particular playback platform. For example, for a single raw streamlet, a single host 504 may produce streamlets for different quality levels for personal computer playback, streamlets for playback on cell phones with a different, proprietary codec, a small video-only streamlet for use when playing just a thumbnail view of the stream (like in a programming guide), and a very high quality streamlet for use in archiving.

[0077] Figure 6a is a schematic block diagram illustrating one embodiment of a virtual timeline 600 in accordance with the present invention. In one embodiment, the virtual timeline 600 comprises at least one quantum media extension 602. The quantum media extension (hereinafter "QMX") 602 describes an entire content file 200. Therefore, the virtual timeline (hereinafter "VT") 600 may comprise a file that is configured to define a playlist for a user to view. For example, the VT may indicate that the publisher desires a user to watch a first show QMX 602a followed by QMX 602b and QMX 602c. As such, the publisher may define a broadcast schedule in a manner similar to a television station.

[0078] Figure 6b is a schematic block diagram illustrating an alternative embodiment of a VT 600 in accordance with the present invention. In the depicted embodiment, the VT 600 may include a single QMX 602 which indicates that the publisher desires the same content to be looped over and over again. For example, the publisher may wish to broadcast a never-ending infomercial on a website.

[0079] Figure 6c is a schematic block diagram illustrating one embodiment of a QMX 602 in accordance with the present invention. In one embodiment, the QMX 602 contains a multitude of information generated by the content module 112 configured to describe the content file 200. Examples of information include, but are not limited to,

start index 604, end index 606, whether the content is live 608, proprietary publisher data 610, encryption level 612, content duration 614 and bitrate values 616. The bitrate values 616 may include frame size 618, audio channel 620 information, codecs 622 used, sample rate 624, and frames parser 626.

[0080] A publisher may utilize the QVT 600 together with the QMX 602 in order to prescribe a playback order for users, or alternatively selectively edit content. For example, a publisher may indicate in the QMX 602 that audio should be muted at time index 10:42 or video should be skipped for 3 seconds at time index 18:35. As such, the publisher may selectively skip offensive content without the processing requirements of editing the content.

[0081] Figure 7 is a schematic block diagram graphically illustrating one embodiment of a client module 114 in accordance with the present invention. The client module 114 may comprise an agent controller module 702, a streamlet cache module 704, and a network controller module 706. In one embodiment, the agent controller module 702 is configured to interface with a viewer 708, and transmit streamlets 304 to the viewer 708. Alternatively, the agent controller module 702 may be configured to simply reassemble streamlets into a single file for transfer to an external device such as a portable video player.

[0082] In a further embodiment, the client module 114 may comprise a plurality of agent controller modules 702. Each agent controller module 702 may be configured to interface with one viewer 708. Alternatively, the agent controller module 702 may be configured to interface with a plurality of viewers 708. The viewer 708 may be a media player (not shown) operating on a PC or handheld electronic device.

[0083] The agent controller module 702 is configured to select a quality level of streamlets to transmit to the viewer 708. The agent controller module 702 requests lower or higher quality streams based upon continuous observation of time intervals between successive receive times of each requested streamlet. The method of requesting higher or lower quality streams will be discussed in greater detail below with reference to Figure 10.

[0084] The agent controller module 702 may be configured to receive user commands from the viewer 708. Such commands may include play, fast forward, rewind, pause, and

stop. In one embodiment, the agent controller module 702 requests streamlets 304 from the streamlet cache module 704 and arranges the received streamlets 304 in a staging module 709. The staging module 709 may be configured to arrange the streamlets 304 in order of ascending playback time. In the depicted embodiment, the streamlets 304 are numbered 0, 1, 2, 3, 4, etc. However, each streamlet 304 may be identified with a unique filename.

[0085] Additionally, the agent controller module 702 may be configured to anticipate streamlet 304 requests and pre-request streamlets 304. By pre-requesting streamlets 304, the user may fast-forward, skip randomly, or rewind through the content and experience no buffering delay. In a further embodiment, the agent controller module 702 may request the streamlets 304 that correspond to time index intervals of 30 seconds within the total play time of the content. Alternatively, the agent controller module 702 may request streamlets at any interval less than the length of the time index. This enables a "fast-start" capability with no buffering wait when starting or fast-forwarding through content file 200. In a further embodiment, the agent controller module 702 may be configured to pre-request streamlets 304 corresponding to specified index points within the content or within other content in anticipation of the end user 104 selecting new content to view. In one embodiment, the streamlet cache module 704 is configured to receive streamlet 304 requests from the agent controller module 702. Upon receiving a request, the streamlet cache module 704 first checks a streamlet cache 710 to verify if the streamlet 304 is present. In a further embodiment, the streamlet cache module 704 handles streamlet 304 requests from a plurality of agent controller modules 702. Alternatively, a streamlet cache module 704 may be provided for each agent controller module 702. If the requested streamlet 304 is not present in the streamlet cache 410, the request is passed to the network controller module 706. In order to enable fast forward and rewind capabilities, the streamlet cache module 704 is configured to store the plurality of streamlets 304 in the streamlet cache 710 for a specified time period after the streamlet 304 has been viewed. However, once the streamlets 304 have been deleted, they may be requested again from the web server 116.

[0086] The network controller module 706 may be configured to receive streamlet requests from the streamlet cache module 704 and open a connection to the web server

116 or other remote streamlet 304 database (not shown). In one embodiment, the network controller module 706 opens a TCP/IP connection to the web server 116 and generates a standard HTTP GET request for the requested streamlet 304. Upon receiving the requested streamlet 304, the network controller module 706 passes the streamlet 304 to the streamlet cache module 704 where it is stored in the streamlet cache 710. In a further embodiment, the network controller module 706 is configured to process and request a plurality of streamlets 304 simultaneously. The network controller module 706 may also be configured to request a plurality of streamlets, where each streamlet 304 is subsequently requested in multiple parts.

[0087] In a further embodiment, streamlet requests may comprise requesting pieces of any streamlet file. Splitting the streamlet 304 into smaller pieces or portions beneficially allows for an increased efficiency potential, and also eliminates problems associated with multiple full-streamlet requests sharing the bandwidth at any given moment. This is achieved by using parallel TCP/IP connections for pieces of the streamlets 304. Consequently, efficiency and network loss problems are overcome, and the streamlets arrive with more useful and predictable timing.

[0088] In one embodiment, the client module 114 is configured to use multiple TCP connections between the client module 114 and the web server 116 or web cache. The intervention of a cache may be transparent to the client or configured by the client as a forward cache. By requesting more than one streamlet 304 at a time in a manner referred to as "parallel retrieval," or more than one part of a streamlet 304 at a time, efficiency is raised significantly and latency is virtually eliminated. In a further embodiment, the client module allows a maximum of three outstanding streamlet 304 requests. The client module 114 may maintain additional open TCP connections as spares to be available should another connection fail. Streamlet 304 requests are rotated among all open connections to keep the TCP flow logic for any particular connection from falling into a slow-start or close mode. If the network controller module 706 has requested a streamlet 304 in multiple parts, with each part requested on mutually independent TCP/IP connections, the network controller module 706 reassembles the parts to present a complete streamlet 304 for use by all other components of the client module 114.

[0089] When a TCP connection fails completely, a new request may be sent on a different connection for the same streamlet 304. In a further embodiment, if a request is not being satisfied in a timely manner, a redundant request may be sent on a different connection for the same streamlet 304. If the first streamlet request's response arrives before the redundant request response, the redundant request can be aborted. If the redundant request response arrives before the first request response, the first request may be aborted.

[0090] Several streamlet 304 requests may be sent on a single TCP connection, and the responses are caused to flow back in matching order along the same connection. This eliminates all but the first request latency. Because multiple responses are always being transmitted, the processing latency of each new streamlet 304 response after the first is not a factor in performance. This technique is known in the industry as "pipelining." Pipelining offers efficiency in request-response processing by eliminating most of the effects of request latency. However, pipelining has serious vulnerabilities. Transmission delays affect all of the responses. If the single TCP connection fails, all of the outstanding requests and responses are lost. Pipelining causes a serial dependency between the requests.

[0091] Multiple TCP connections may be opened between the client module 114 and the web server 116 to achieve the latency-reduction efficiency benefits of pipelining while maintaining the independence of each streamlet 304 request. Several streamlet 304 requests may be sent concurrently, with each request being sent on a mutually distinct TCP connection. This technique is labeled "virtual pipelining" and is an innovation of the present invention. Multiple responses may be in transit concurrently, assuring that communication bandwidth between the client module 114 and the web server 116 is always being utilized. Virtual pipelining eliminates the vulnerabilities of traditional pipelining. A delay in or complete failure of one response does not affect the transmission of other responses because each response occupies an independent TCP connection. Any transmission bandwidth not in use by one of multiple responses (whether due to delays or TCP connection failure) may be utilized by other outstanding responses.

[0092] A single streamlet 304 request may be issued for an entire streamlet 304, or multiple requests may be issued, each for a different part or portion of the streamlet. If the streamlet is requested in several parts, the parts may be recombined by the client module 114 streamlet.

[0093] In order to maintain a proper balance between maximized bandwidth utilization and response time, the issuance of new streamlet requests must be timed such that the web server 116 does not transmit the response before the client module 114 has fully received a response to one of the previously outstanding streamlet requests. For example, if three streamlet 304 requests are outstanding, the client module 114 should issue the next request slightly before one of the three responses is fully received and "out of the pipe." In other words, request timing is adjusted to keep three responses in transit. Sharing of bandwidth among four responses diminishes the net response time of the other three responses. The timing adjustment may be calculated dynamically by observation, and the request timing adjusted accordingly to maintain the proper balance of efficiency and response times.

[0094] The schematic flow chart diagrams that follow are generally set forth as logical flow chart diagrams. As such, the depicted order and labeled steps are indicative of one embodiment of the presented method. Other steps and methods may be conceived that are equivalent in function, logic, or effect to one or more steps, or portions thereof, of the illustrated method. Additionally, the format and symbols employed are provided to explain the logical steps of the method and are understood not to limit the scope of the method. Although various arrow types and line types may be employed in the flow chart diagrams, they are understood not to limit the scope of the corresponding method. Indeed, some arrows or other connectors may be used to indicate only the logical flow of the method. For instance, an arrow may indicate a waiting or monitoring period of unspecified duration between enumerated steps of the depicted method. Additionally, the order in which a particular method occurs may or may not strictly adhere to the order of the corresponding steps shown.

[0095] Figure 8 is a schematic flow chart diagram illustrating one embodiment of a method 800 for processing content in accordance with the present invention. In one embodiment the method 800 starts 802, and the content module 112 receives 804 content

from the publisher 110. Receiving content 804 may comprise receiving 804 a digital copy of the content file 200, or digitizing a physical copy of the content file 200. Alternatively, receiving 804 content may comprise capturing a radio, television, cable, or satellite broadcast. Once received 804, the streamlet module 404 generates 808 a plurality of source streamlets 303 each having a fixed duration. Alternatively, the streamlets 303 may be generated with a fixed file size.

[0096] In one embodiment, generating 808 streamlets comprises dividing the content file 200 into a plurality of two second streamlets 303. Alternatively, the streamlets may have any length less than or equal to the length of the stream 202. The encoder module 406 then encodes 810 the streamlets 303 into sets 306 of streamlets 304, in a plurality of streams 202 according to an encoding scheme. The quality may be predefined, or automatically set according to end user bandwidth, or in response to pre-designated publisher guidelines

[0097] In a further embodiment, the encoding scheme comprises a proprietary codec such as WMV9®. The encoder module 406 then stores 812 the encoded streamlets 304 in the streamlet database 408. Once stored 812, the web server 116 may then serve 814 the streamlets 304. In one embodiment, serving 814 the streamlets 304 comprises receiving streamlet requests from the client module 114, retrieving the requested streamlet 304 from the streamlet database 408, and subsequently transmitting the streamlet 304 to the client module 114. The method 800 then ends 816.

[0098] Figure 9 is a schematic flow chart diagram illustrating one embodiment of a method 900 for viewing a plurality of streamlets in accordance with the present invention. The method 900 starts and an agent controller module 702 is provided 904 and associated with a viewer 708 and provided with a staging module 709. The agent controller module 702 then requests 906 a streamlet 304 from the streamlet cache module 704. Alternatively, the agent controller module 702 may simultaneously request 906 a plurality of streamlets 304 the streamlet cache module 704. If the streamlet is stored 908 locally in the streamlet cache 710, the streamlet cache module 704 retrieves 910 the streamlet 304 and sends the streamlet to the agent controller module 702. Upon retrieving 910 or receiving a streamlet, the agent controller module 702 makes 911 a

determination of whether or not to shift to a higher or lower quality stream 202. This determination will be described below in greater detail with reference to Figure 10.

[0099] In one embodiment, the staging module 709 then arranges 912 the streamlets 304 into the proper order, and the agent controller module 702 delivers 914 the streamlets to the viewer 708. In a further embodiment, delivering 914 streamlets 304 to the end user comprises playing video and or audio streamlets on the viewer 708. If the streamlets 304 are not stored 908 locally, the streamlet request is passed to the network controller module 706. The network controller module 706 then requests 916 the streamlet 304 from the web server 116. Once the streamlet 304 is received, the network controller module 706 passes the streamlet to the streamlet cache module 704. The streamlet cache module 704 archives 918 the streamlet. Alternatively, the streamlet cache module 704 then archives 918 the streamlet and passes the streamlet to the agent controller module 702, and the method 900 then continues from operation 910 as described above.

[0100] Referring now to Figure 10, shown therein is a schematic flow chart diagram illustrating one embodiment of a method 1000 for requesting streamlets 304 within an adaptive-rate shifting content streaming environment in accordance with the present invention. The method 1000 may be used in one embodiment as the operation 911 of Figure 9. The method 1000 starts and the agent controller module 702 receives 1004 a streamlet 304 as described above with reference to Figure 9. The agent controller module 702 then monitors 1006 the receive time of the requested streamlet. In one embodiment, the agent controller module 702 monitors the time intervals Δ between successive receive times for each streamlet response. Ordering of the responses in relation to the order of their corresponding requests is not relevant.

[0101] Because network behavioral characteristics fluctuate, sometimes quite suddenly, any given Δ may vary substantially from another. In order to compensate for this fluctuation, the agent controller module 702 calculates 1008 a performance ratio r across a window of n samples for streamlets of playback length S. In one embodiment, the performance ratio r is calculated using the equation:

$$r = S \frac{n}{\sum_{i=1}^{n} \Delta_i}$$

[0102] Due to multiple simultaneous streamlet processing, and in order to better judge the central tendency of the performance ratio r, the agent controller module 702 may calculate a geometric mean, or alternatively an equivalent averaging algorithm, across a window of size m, and obtain a performance factor φ:

$$\varphi_{current} = \left(\prod_{j=1}^{m} r_j \right)^{\frac{1}{m}}$$

[0103] The policy determination about whether or not to upshift 1010 playback quality begins by comparing $\varphi_{current}$ with a trigger threshold Θ_{up}. If $\varphi_{current} \geq \Theta_{up}$, then an up shift to the next higher quality stream may be considered 1016. In one embodiment, the trigger threshold Θ_{up} is determined by a combination of factors relating to the current read ahead margin (i.e. the amount of contiguously available streamlets that have been sequentially arranged by the staging module 709 for presentation at the current playback time index), and a minimum safety margin. In one embodiment, the minimum safety margin may be 24 seconds. The smaller the read ahead margin, the larger Θ_{up} is to discourage upshifting until a larger read ahead margin may be established to withstand network disruptions. If the agent controller module 702 is able to sustain 1016 upshift quality, then the agent controller module 702 will upshift 1017 the quality and subsequently request higher quality streams. The determination of whether use of the higher quality stream is sustainable 1016 is made by comparing an estimate of the higher quality stream's performance factor, φ_{higher}, with Θ_{up}. If $\varphi_{higher} \geq \Theta_{up}$ then use of the higher quality stream is considered sustainable. If the decision of whether or not the higher stream rate is sustainable 1016 is "no," the agent controller module 702 will not attempt to upshift 1017 stream quality. If the end of the stream has been reached 1014, the method 1000 ends 1016.

[0104] If the decision on whether or not to attempt upshift 1010 is "no", a decision about whether or not to downshift 1012 is made. In one embodiment, a trigger threshold Θ_{down} is defined in a manner analogous to Θ_{up}. If $\varphi_{current} > \Theta_{down}$ then the stream quality may be adequate, and the agent controller module 702 does not downshift 1018 stream quality.

However, if $\varphi_{current} \leq \Theta_{down}$, the agent controller module 702 does downshift 1018 the stream quality. If the end of the stream has not been reached 1014, the agent controller module 702 begins to request and receive 1004 lower quality streamlets and the method 1000 starts again. Of course, the above described equations and algorithms are illustrative only, and may be replaced by alternative streamlet monitoring solutions.

[0105] The present invention may be embodied in other specific forms without departing from its spirit or essential characteristics. The described embodiments are to be considered in all respects only as illustrative and not restrictive. The scope of the invention is, therefore, indicated by the appended claims rather than by the foregoing description. All changes which come within the meaning and range of equivalency of the claims are to be embraced within their scope.

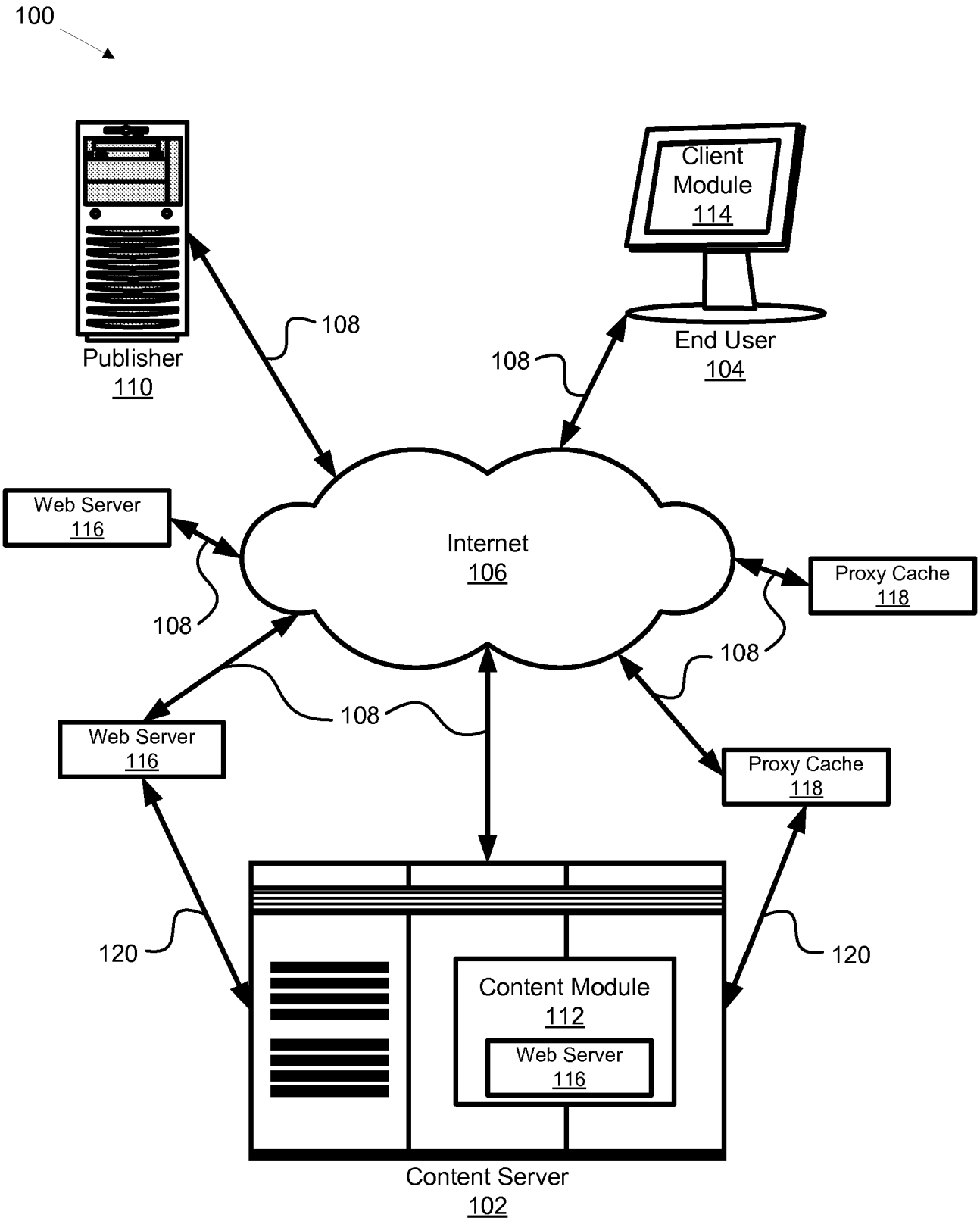

100
Publisher
110
108
Client
Module
114
End User
104
108
Web Server
116
Internet
106
Proxy Cache
118
108
108
108
Web Server
116
Proxy Cache
118
120
120
Content Module
112
Web Server
116
Content Server
102

FIG. 1

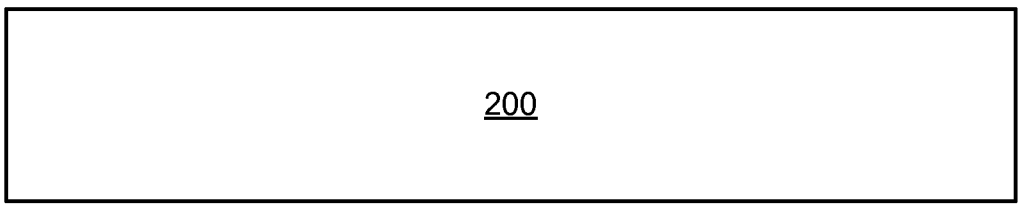


FIG. 2a

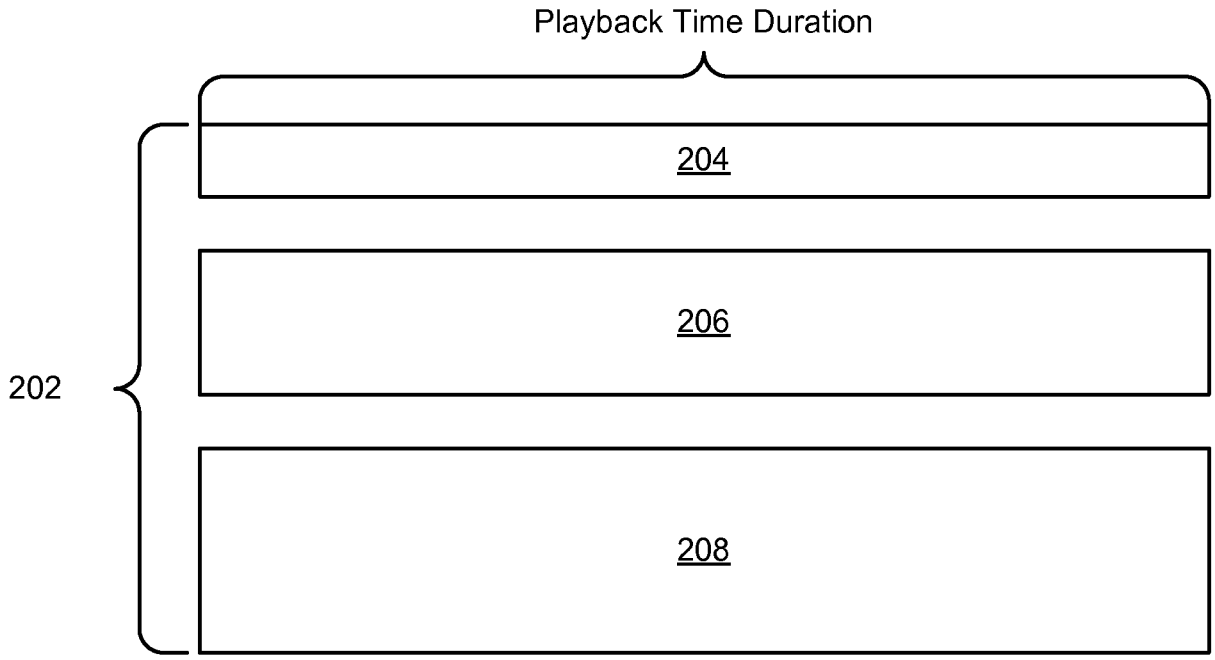


FIG. 2b

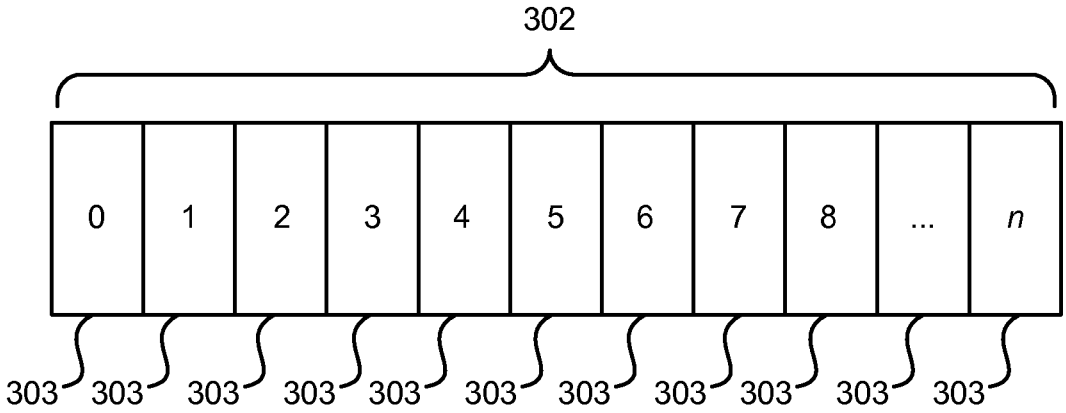

302
0
1
2
3
4
5
6
7
8
...
n
303
303
303
303
303
303
303
303
303
303
303

FIG. 3a

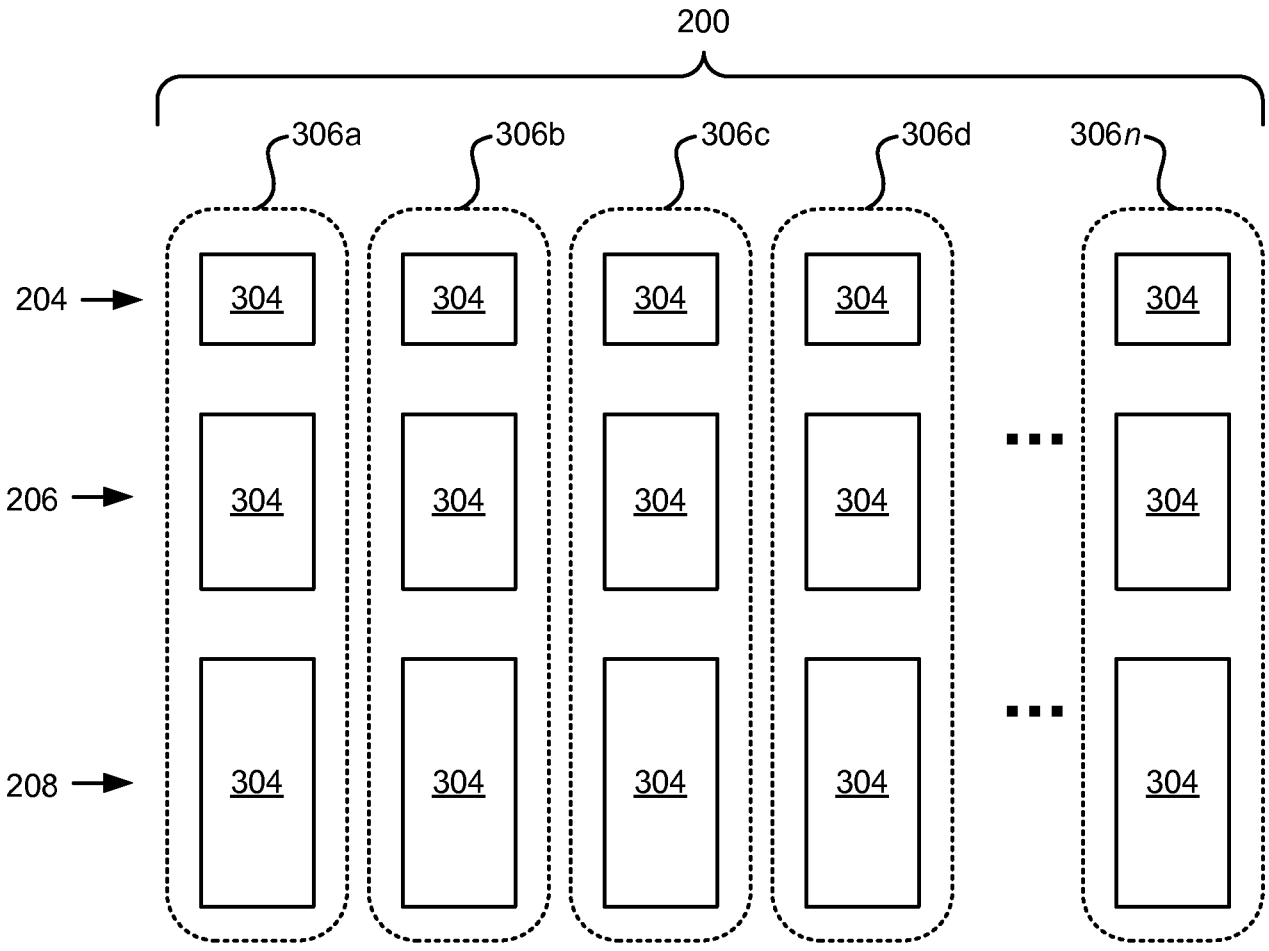

200
306a
306b
306c
306d
306n
204
206
208
304

FIG. 3b

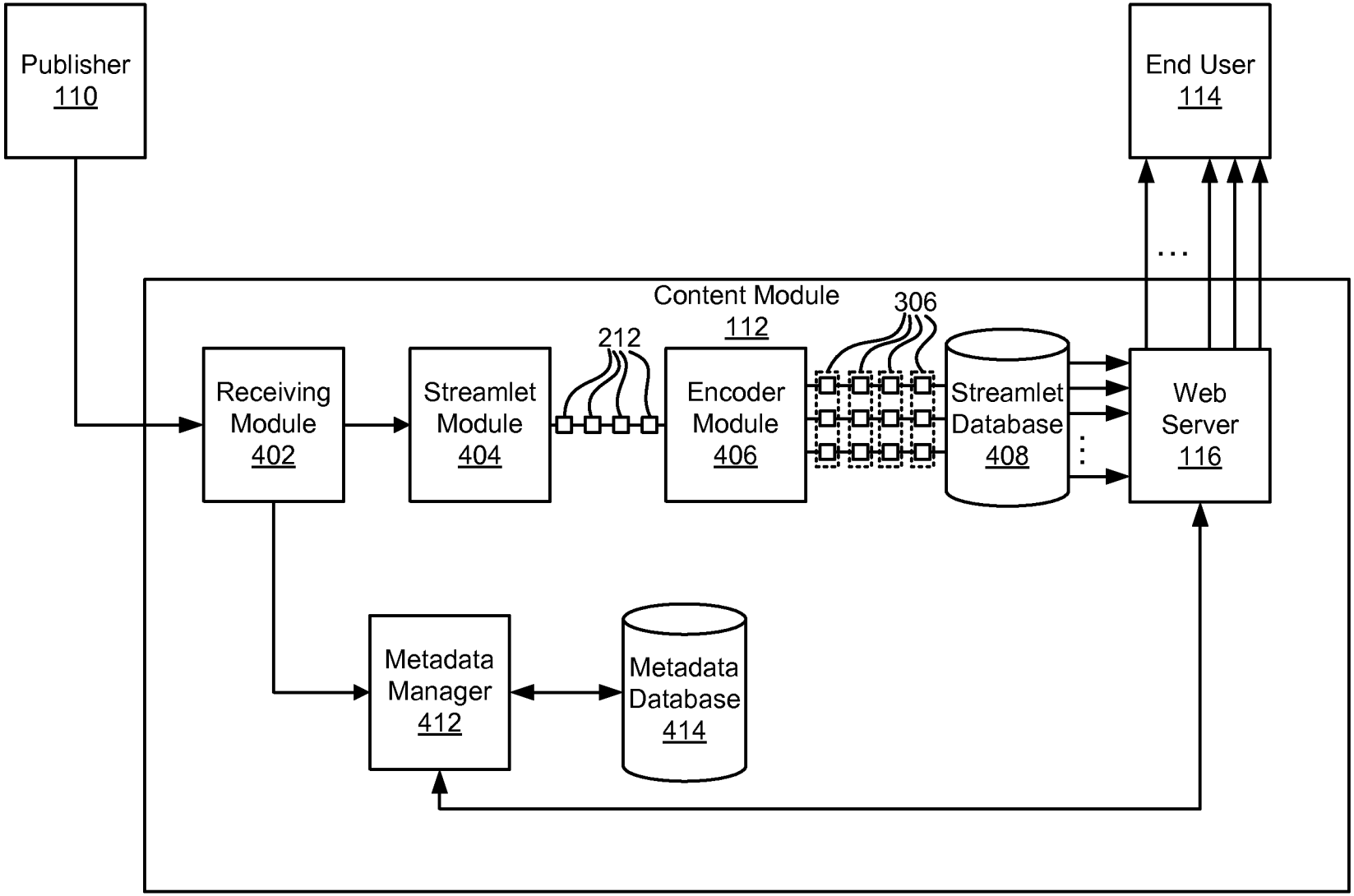

Publisher
110
End User
114
Content Module
112
306
212
Receiving
Module
402
Streamlet
Module
404
Encoder
Module
406
Streamlet
Database
408
Web
Server
116
Metadata
Manager
412
Metadata
Database
414

FIG. 4

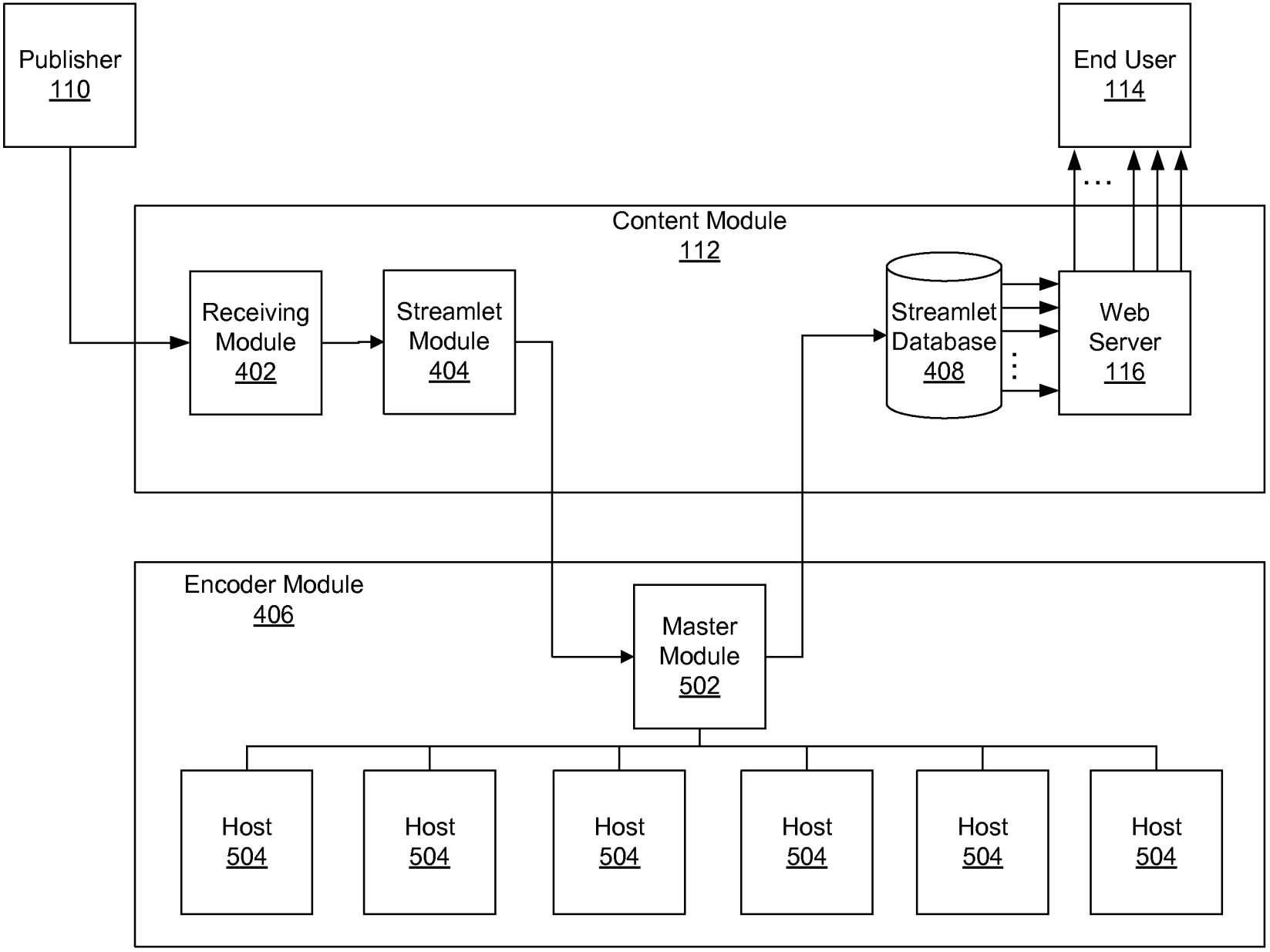

Publisher
110
End User
114
Content Module
112
Receiving
Module
402
Streamlet
Module
404
Streamlet
Database
408
Web
Server
116
Encoder Module
406
Master
Module
502
Host
504
Host
504
Host
504
Host
504
Host
504
Host
504

FIG. 5a

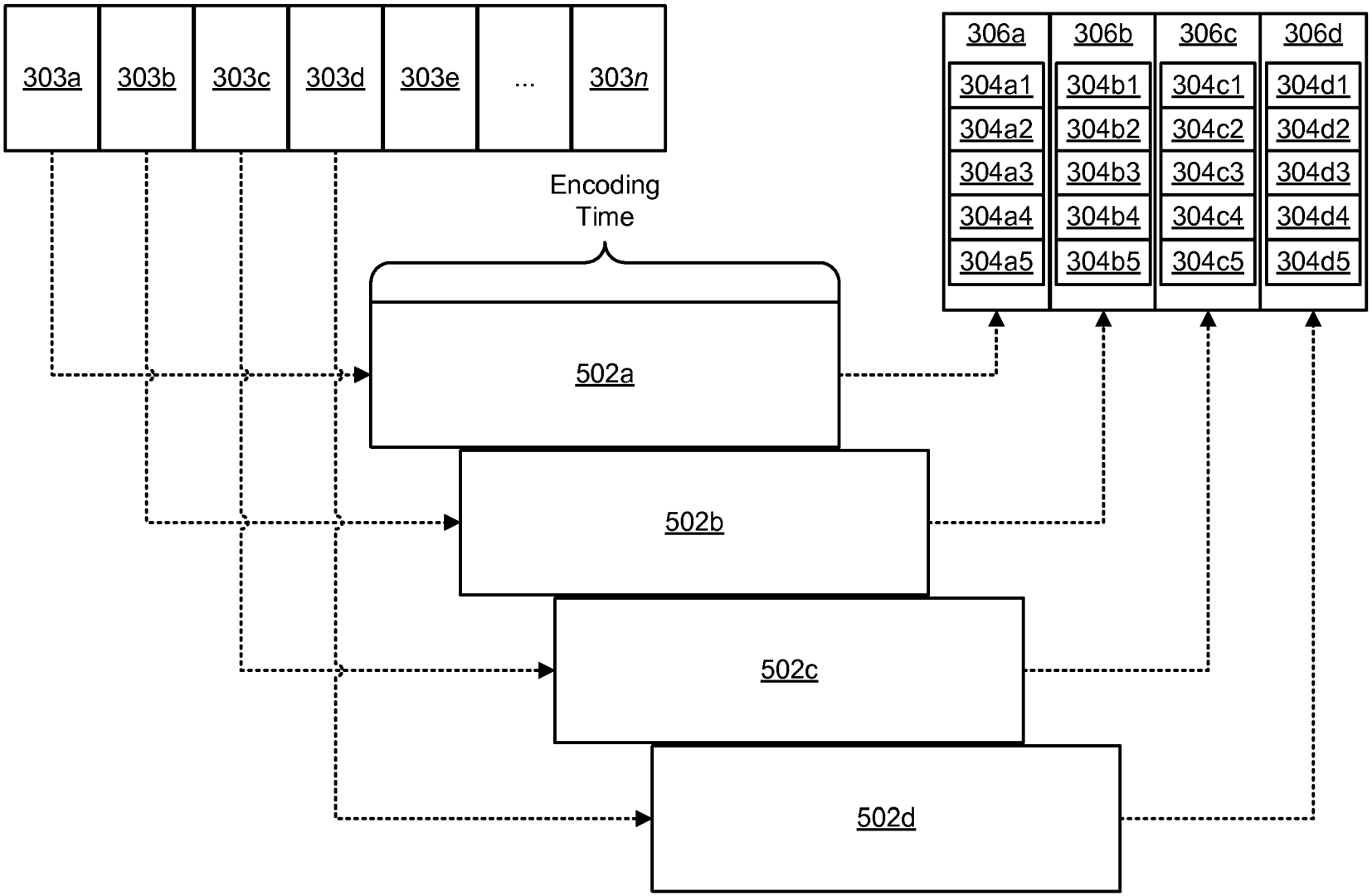

303a
303b
303c
303d
303e
...
303n
Encoding Time
502a
502b
502c
502d
306a
306b
306c
306d
304a1
304a2
304a3
304a4
304a5
304b1
304b2
304b3
304b4
304b5
304c1
304c2
304c3
304c4
304c5
304d1
304d2
304d3
304d4
304d5

FIG. 5b

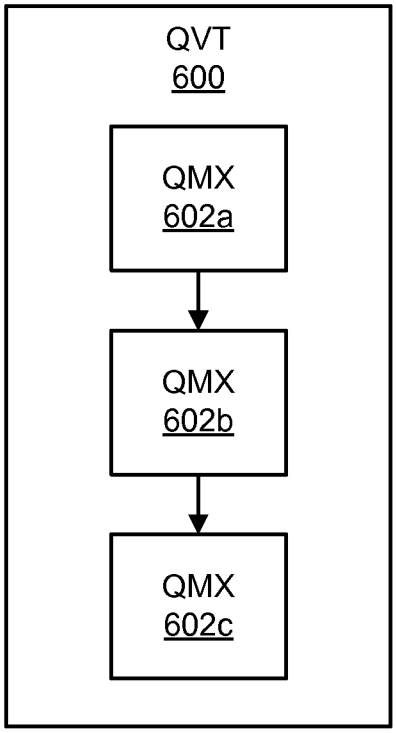


FIG. 6a

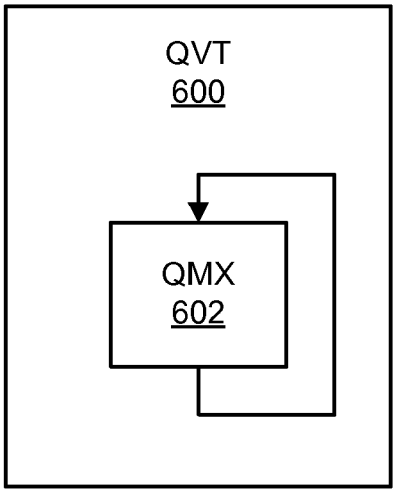


FIG. 6b

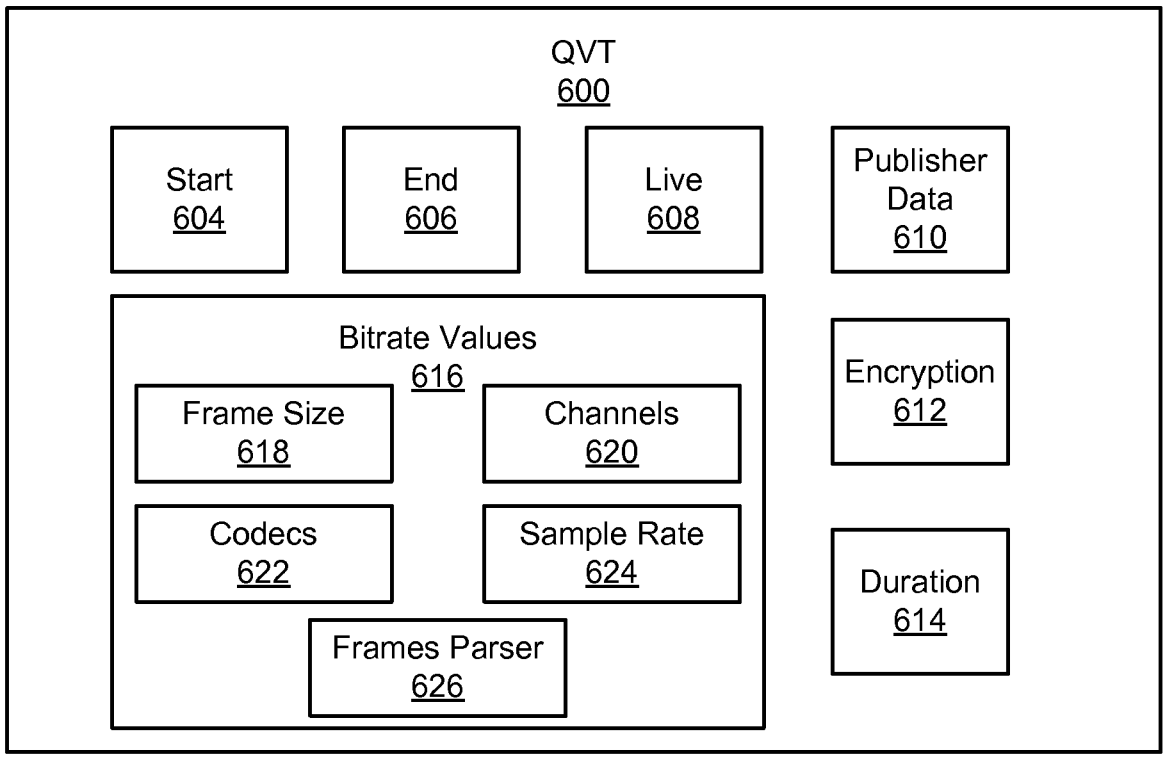


FIG. 6c

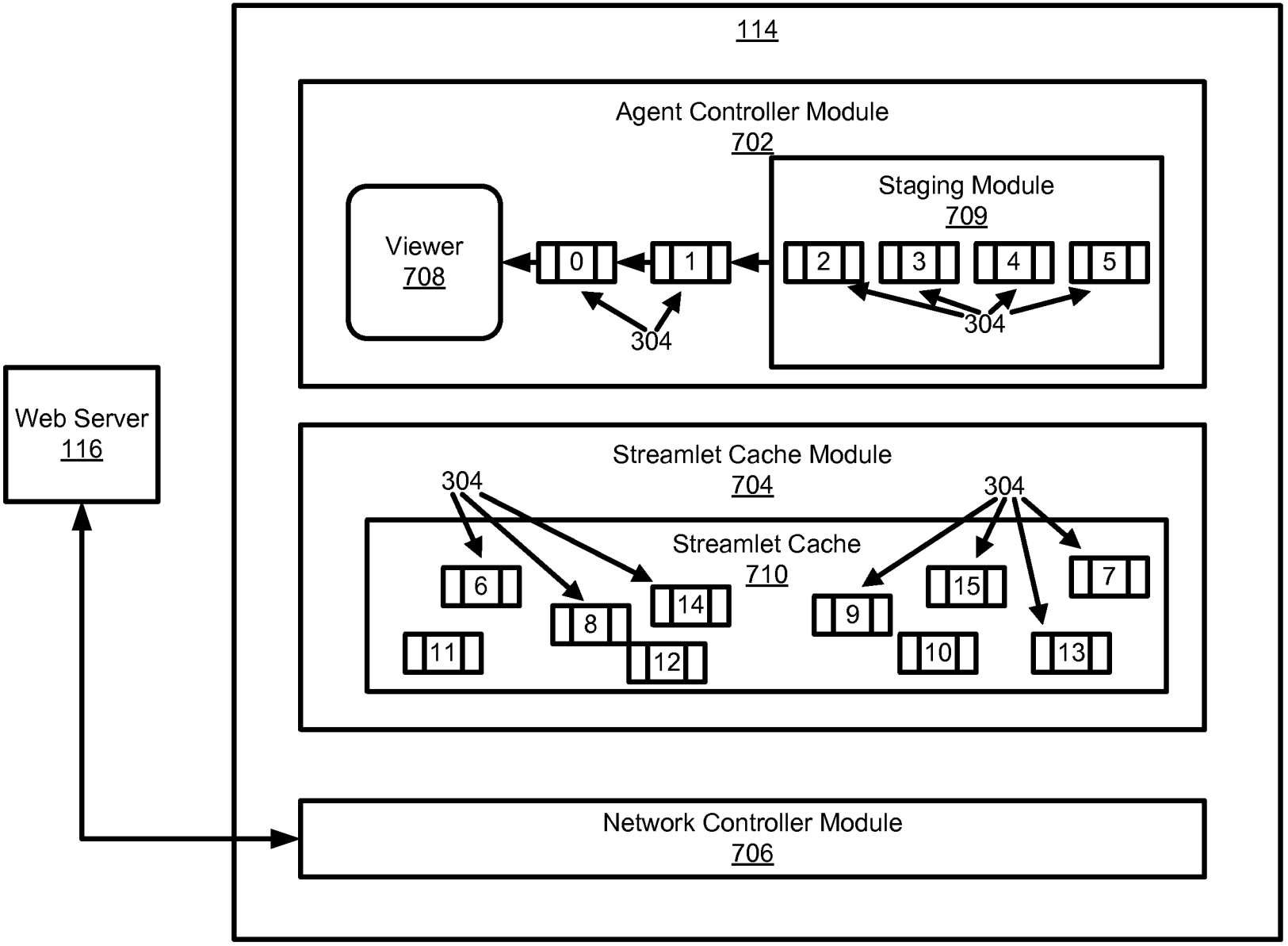

114
Agent Controller Module
702
Staging Module
709
Viewer
708
0
1
2
3
4
5
304
304
Web Server
116
Streamlet Cache Module
704
304
304
Streamlet Cache
710
6
14
8
12
11
9
15
7
10
13
Network Controller Module
706

FIG. 7

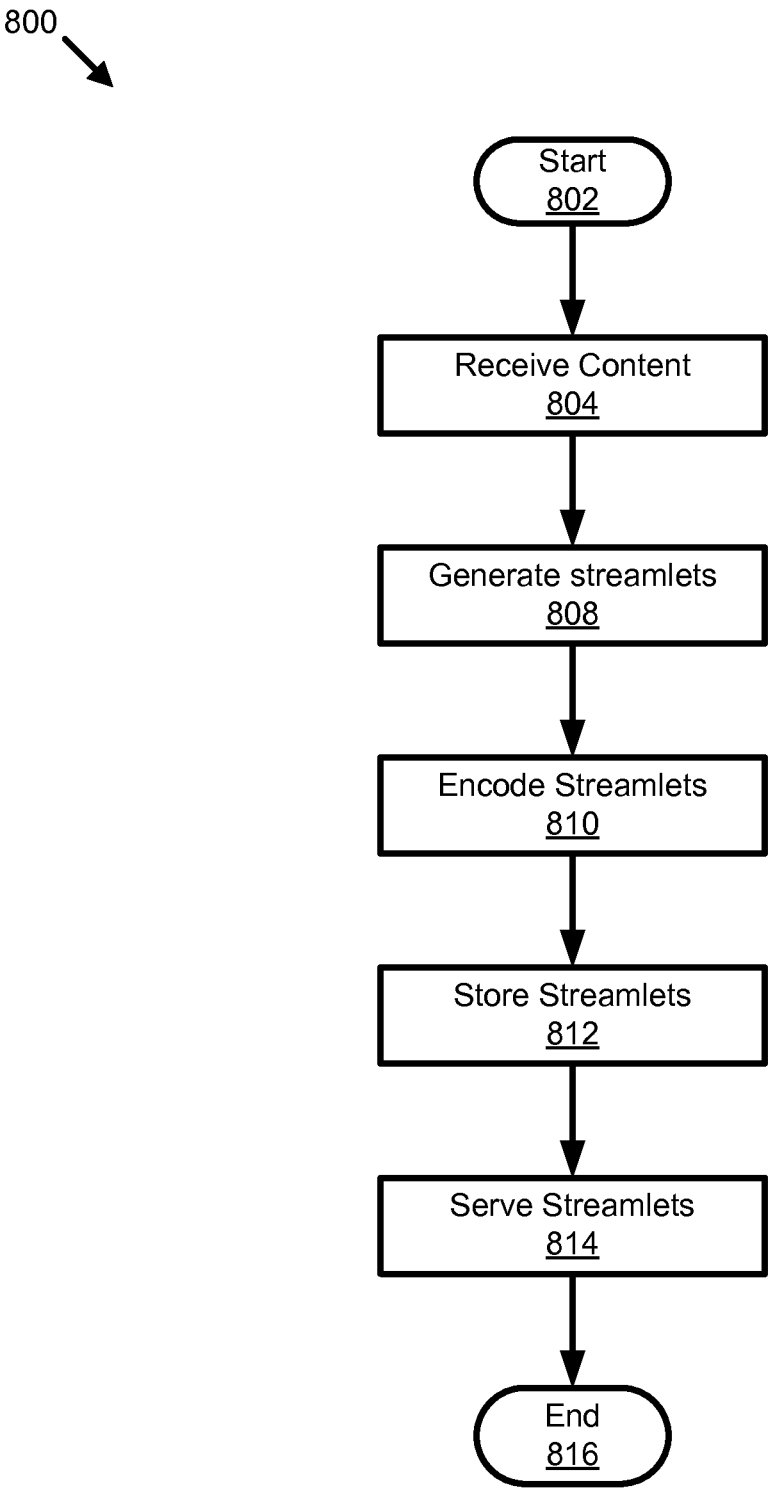

800
Start
802
Receive Content
804
Generate streamlets
808
Encode Streamlets
810
Store Streamlets
812
Serve Streamlets
814
End
816

FIG. 8

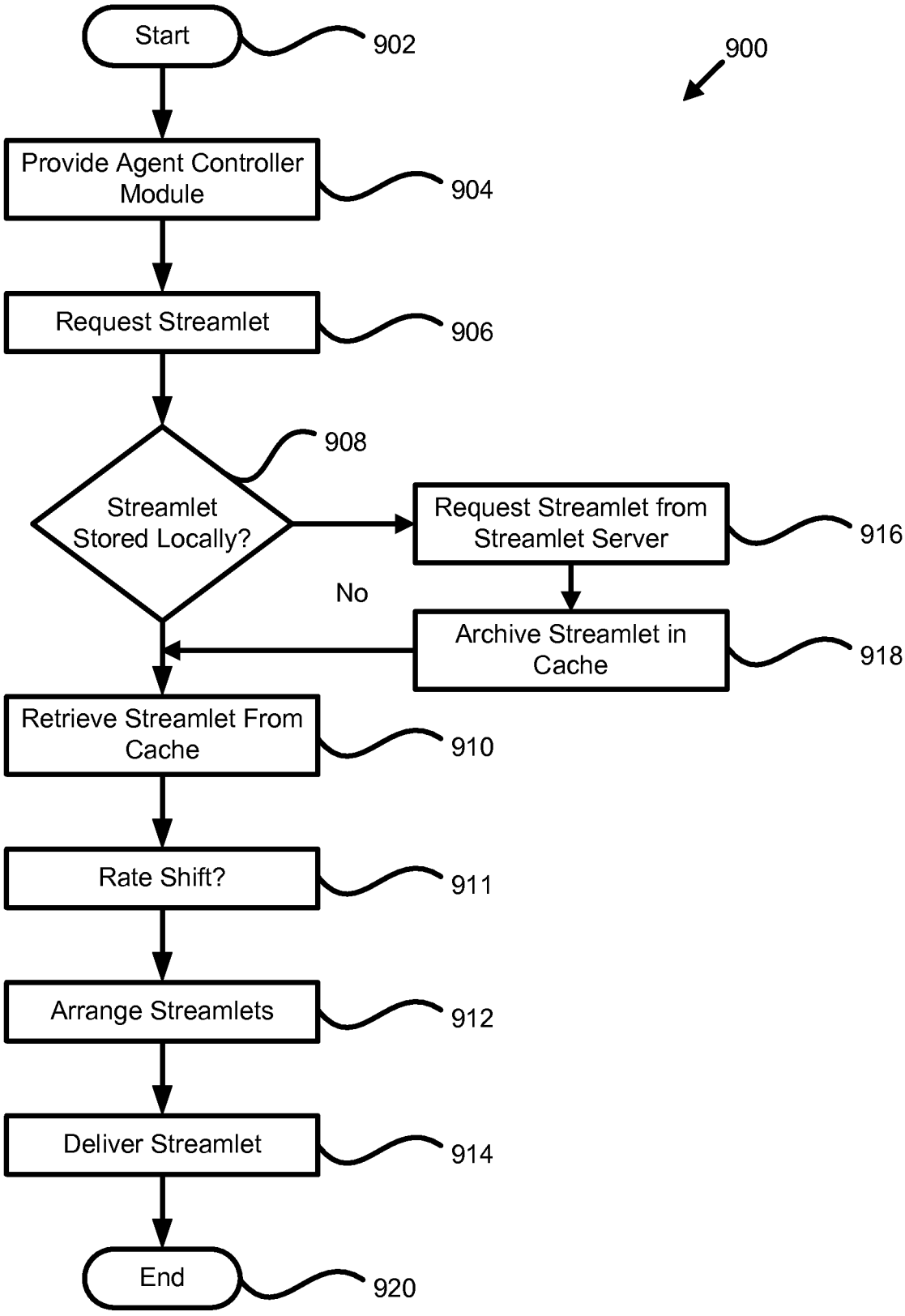

Start
902
900
Provide Agent Controller Module
904
Request Streamlet
906
908
Streamlet Stored Locally?
Request Streamlet from Streamlet Server
916
No
Archive Streamlet in Cache
918
Retrieve Streamlet From Cache
910
Rate Shift?
911
Arrange Streamlets
912
Deliver Streamlet
914
End
920

FIG. 9

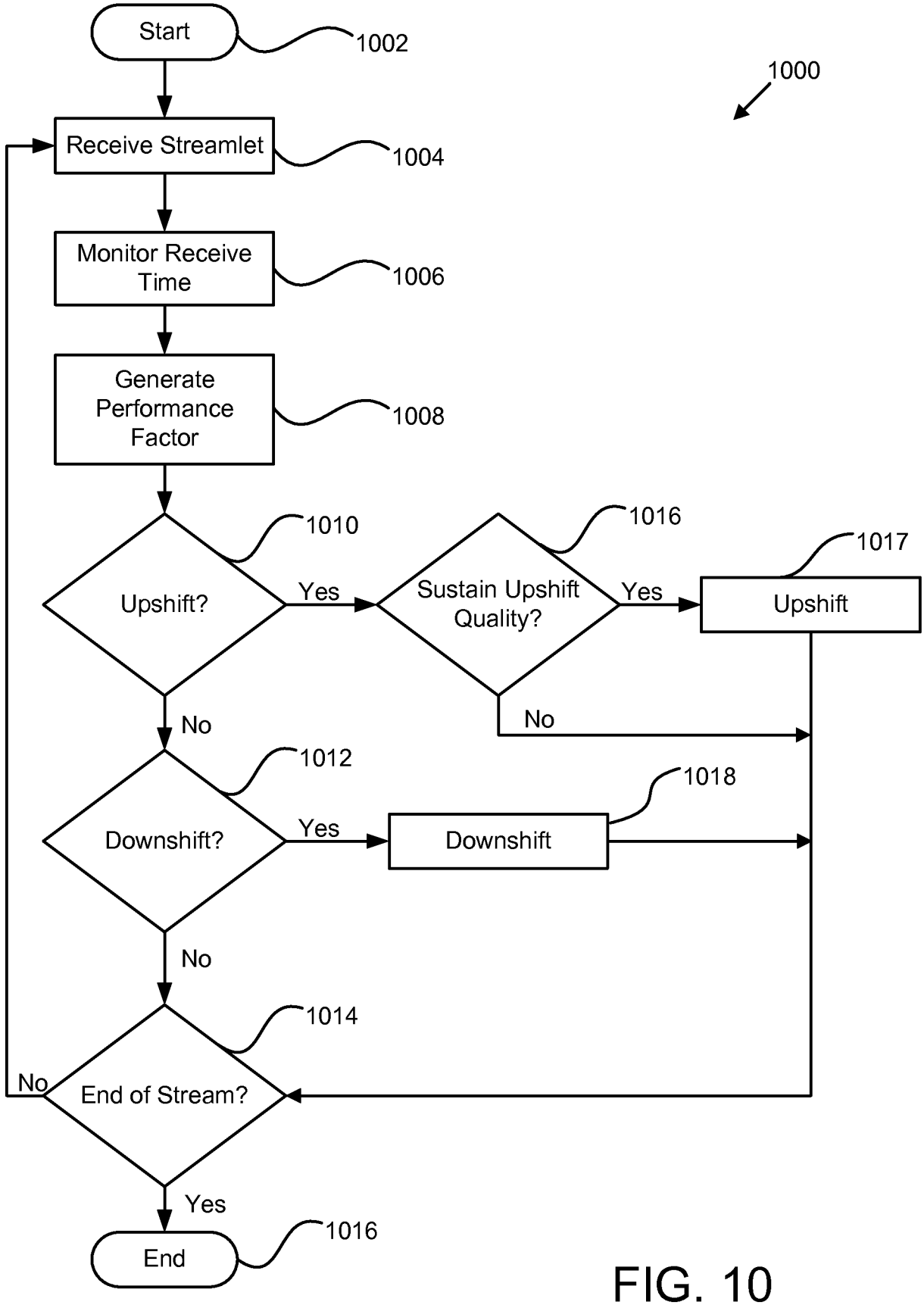

Start
1002
1000
Receive Streamlet
1004
Monitor Receive Time
1006
Generate Performance Factor
1008
1010
Upshift?
Yes
1016
Sustain Upshift Quality?
Yes
1017
Upshift
No
No
1012
Downshift?
Yes
1018
Downshift
No
1014
No
End of Stream?
Yes
1016
End

FIG. 10

DECLARATION OF JOINT INVENTORS RELATING TO PATENTS AND APPLICATIONS ENTITLED "APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING"

This is a joint declaration and statement made by David F. Brueck, Mark B. Hurst, and R. Drew Major. We the undersigned each hereby declare and state that, to the best of our information and belief:

1. Each of us (David F. Brueck, Mark B. Hurst, and R. Drew Major) contributed to the inventorship of the following United States Patents and the following pending application:

US Patents:	US Patent Application
US7818444	Serial No. 15/414,025
US8402156	
US8612624	
US9071668	
US9571551	

2. The above-identified patents and applications were made or authorized to be made by me. I believe that I am the inventor or an original joint inventor of a claimed invention in each of the applications and patents referenced above.

3. Although R. Drew Major was initially omitted as an inventor, I understand that he is now being added as a joint inventor. I agree to the change of inventorship and I have no disagreement with regard to the change in inventorship. To the best of my information and belief, Mr. Major's earlier omission was made without deceptive intent.

4. I hereby declare that all statements made herein of my own knowledge are true and that all statements on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements in this declaration are punishable under 18 U.S.C. 1001 by fine or imprisonment of not more than five (5) years, or both, and

that willful false statements may jeopardize the validity of the application or any patent issued thereon.

Respectfully submitted by:

David F. Brueck — Date: 27 Oct 2017

Mark B. Hurst — Date:

R. Drew Major — Date:

1001 by fine or imprisonment of not more than five (5) years, or both, and that willful false statements may jeopardize the validity of the application or any patent issued thereon.

Respectfully submitted by:

David F. Brueck	Date
Mark B. Hurst	11/6/17 Date
R. Drew Major	11/6/17 Date

CLAIMS

What is claimed is:

1. A system for adaptive-rate content streaming of video playable on one or more end user stations over the Internet, the system comprising:

at least one storage device storing video, the video encoded at a plurality of different bitrates creating a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, the low quality stream, the medium quality stream, and the high quality stream each comprising a group of streamlets encoded at a respective one of the plurality of different bitrates, each group of streamlets comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and

wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the video in each of the low quality stream, the medium quality stream, and the high quality stream, and wherein the first streamlet of the low quality stream encodes the same first portion of the video at a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream.

2. The system of claim 1, wherein the video is a live event video.

3. The system of claim 1, wherein the video includes archived content.

4. The system of claim 1, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

5. The system of claim 4, wherein the first and second durations are different.

6. The system of claim 1, further comprising: a plurality of web servers located at different locations across the internet, each web server configured to: receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing a portion of the video, wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams; retrieve from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and send the retrieved first streamlet from the currently selected one of the different copies to the requesting one of the end user stations over the one or more network connections.

7. The system of claim 1, wherein each of the first streamlets has a first duration that is the range of 0.1 to 5 seconds.

8. The system of claim 1, further comprising:

a first web server configured to: receive at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the video,

wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the video; retrieve from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and send the retrieved first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream to the requesting one of the end user stations over the one or more network connections.

9. The system of claim 8, wherein the first web server is further configured to: receive at least one virtual timeline request over the one or more internet connections from the one or more end user stations to retrieve a virtual timeline; and send the virtual timeline to the requesting one of the end user stations over the one or more network connections.

10. The system of claim 9, wherein the virtual timeline corresponds to the currently selected one of the low quality stream, the medium quality stream, and the high quality stream.

11. The system of claim 10, wherein the virtual timeline defines a playlist for a user to view.

12. The system of claim 11, wherein the virtual timeline comprises a file that is configured to define a playlist for a user to view.

13. The system of claim 12, wherein the virtual timeline comprises at least one quantum media extension (QMX).

14. An end user station to stream a video over a network from a server for playback of the video, the content player device comprising:

a processor;

a digital processing apparatus memory device comprising non-transitory machine-readable instructions that, when executed, cause the processor to:

establish one or more network connections between the end user station and the server, wherein the server is configured to access at least one of a plurality of groups of streamlets;

wherein the video is encoded at a plurality of different bitrates to create a plurality of streams including at least a low quality stream, a medium quality stream, and a high quality stream, each of the low quality stream, the medium quality stream, and the high quality stream comprising a group of streamlets encoded at the same respective one of the different bitrates, each group

comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bit rate of no less than 600 kbps; and wherein the first streamlets of each of the low quality stream, the medium quality stream and the high quality stream each has an equal playback duration and each of the first streamlets encodes the same portion of the video at a different one of the different bitrates;

select a specific one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the streams;

place a streamlet request to the server over the one or more network connections for the first streamlet of the selected stream; receive the requested first streamlet from the server via the one or more network connections; and

provide the received first streamlet for playback of the video.

15. The end user station of claim 14, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

16. The end user station of claim 15, wherein the first and second durations are different.

17. The end user station of claim 14, wherein each of the first streamlets has a first duration that is the range of 0.1 to 5 seconds.

18. The end user station of claim 14, wherein the end user station is further configured to: request and receive a virtual timeline; and wherein one or more streamlet requests are based on the at least one virtual timeline.

19. The end user station of claim 18, wherein the virtual timeline corresponds to the currently selected one of the low quality stream, the medium quality stream, and the high quality stream.

20. The end user station of claim 18, wherein the virtual timeline defines a playlist for a user to view.

21. The end user station of claim 14, wherein the video is a live event video.

22. The end user station of claim 14, wherein the video includes archived content.

23. A process executable by one or more servers to stream a video for playback by one or more end user stations, the process comprising:

storing, by the one or more servers, a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video;

wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and

wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the first streamlet of the low quality stream having a different one of the different bitrates than the first streamlet of the high quality stream and the first streamlet of the medium quality stream;

receiving at least one streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the video,

wherein the at least one streamlet request from the one or more end user stations includes a request for a currently selected first streamlet from one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the video;

retrieving from the storage device the requested first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream; and

sending the retrieved first streamlet from the currently selected one of the low quality stream, the medium quality stream, and the high quality stream to the requesting one of the end user stations over the one or more network connections.

24. The process of claim 23, wherein the second streamlet of each of the groups of streamlets each has the same second duration and corresponds to the same second portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the second streamlet of the low quality stream having the same bitrate as the first streamlet of the low quality stream.

25. The process of claim 23, wherein the first and second durations are different.

26. The process of claim 23, wherein the video is a live event video.

27. The process of claim 23, wherein the video includes archived content.

28. A process executable by a content player device to stream a video over a network from a server for playback of the video by the content player device, the process comprising:

establishing one or more network connections between the content player device and the server,

wherein the server accesses a plurality of streams including a low quality stream, a medium quality stream, and a high quality stream, wherein the low

quality stream, the medium quality stream, and the high quality stream each comprise a group of streamlets encoded at a respective one of a plurality of different bitrates, each group comprising at least first and second streamlets, each of the streamlets corresponding to a portion of the video; wherein at least one of the low quality stream, the medium quality stream, and the high quality stream is encoded at a bitrate of no less than 600 kbps; and

wherein the first streamlet of each of the groups of streamlets has the same first duration and encodes the same first portion of the video in the low quality stream, the medium quality stream, and the high quality stream, the first streamlet of the low quality stream having a different bitrate than the first streamlet of the high quality stream and the first streamlet of the medium quality stream;

selecting, by the content player device, a currently selected one of the low quality stream, the medium quality stream, and the high quality stream based upon a determination by the end user station to select a higher or lower bitrate version of the video;

placing a streamlet request over one or more internet connections from the one or more end user stations to retrieve the first streamlet storing the first portion of the video; and

receiving the requested streamlet from the server via the one or more network connections; and rendering, by the content player device, the received streamlet for playback of the video.

29. The process of claim 28, wherein the video includes archived content.

30. The process of claim 28, wherein the video is a live event video.

ABSTRACT OF THE DISCLOSURE

An apparatus for multi-bitrate content streaming includes a receiving module configured to capture media content, a streamlet module configured to segment the media content and generate a plurality of streamlets, and an encoding module configured to generate a set of streamlets. The system includes the apparatus, wherein the set of streamlets comprises a plurality of streamlets having identical time indices and durations, and each streamlet of the set of streamlets having a unique bitrate, and wherein the encoding module comprises a master module configured to assign an encoding job to one of a plurality of host computing modules in response to an encoding job completion bid. A method includes receiving media content, segmenting the media content and generating a plurality of streamlets, and generating a set of streamlets.

Aylo Exhibit 1002
Page 61 of 1405

Electronic Patent Application Fee Transmittal

Application Number:	
Filing Date:	
Title of Invention:	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING
First Named Inventor/Applicant Name:	DAVID F. BRUECK
Filer:	Brett Carlson/Julie Graf
Attorney Docket Number:	066.0141X1C9

Filed as Large Entity

Filing Fees for Track I Prioritized Examination - Nonprovisional Application under 35 USC 111(a)

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Basic Filing:				
UTILITY APPLICATION FILING	1011	1	300	300
UTILITY SEARCH FEE	1111	1	660	660
UTILITY EXAMINATION FEE	1311	1	760	760
REQUEST FOR PRIORITIZED EXAMINATION	1817	1	4000	4000
Pages:				
Claims:				
CLAIMS IN EXCESS OF 20	1202	10	100	1000
INDEPENDENT CLAIMS IN EXCESS OF 3	1201	1	460	460

Description	Fee Code	Quantity	Amount	Sub-Total in USD($)
Miscellaneous-Filing:				
PUBL. FEE- EARLY, VOLUNTARY, OR NORMAL	1504	1	0	0
PROCESSING FEE, EXCEPT PROV. APPLS.	1830	1	140	140
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				
Miscellaneous:				
		Total in USD ($)		**7320**

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
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.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

The application data sheet is part of the provisional or nonprovisional application for which it is being submitted. The following form contains the bibliographic data arranged in a format specified by the United States Patent and Trademark Office as outlined in 37 CFR 1.76.
This document may be completed electronically and submitted to the Office in electronic format using the Electronic Filing System (EFS) or the document may be printed and included in a paper filed application.

Secrecy Order 37 CFR 5.2:

☐ Portions or all of the application associated with this Application Data Sheet may fall under a Secrecy Order pursuant to 37 CFR 5.2 (Paper filers only. Applications that fall under Secrecy Order may not be filed electronically.)

Inventor Information:

Inventor 1 Remove

Legal Name

Prefix	Given Name	Middle Name	Family Name	Suffix
	DAVID	F.	BRUECK	

Residence Information (Select One) ● US Residency Non US Residency Active US Military Service

City	State/Province	Country of Residence
SARATOGA SPRINGS	UT	US

Mailing Address of Inventor:

Address 1	962 NORTH RATTLER ROAD
Address 2	

City	State/Province	Postal Code	Country
SARATOGA SPRINGS	UT	84043	US

Inventor 2 Remove

Legal Name

Prefix	Given Name	Middle Name	Family Name	Suffix
	MARK	B.	HURST	

Residence Information (Select One) ◉ US Residency Non US Residency Active US Military Service

City	State/Province	Country of Residence
CEDAR HILLS	UT	US

Mailing Address of Inventor:

Address 1	10639 NORTH AVALON
Address 2	

City	State/Province	Postal Code	Country
CEDAR HILLS	UT	84062	US

Inventor 3 Remove

Legal Name

Prefix	Given Name	Middle Name	Family Name	Suffix
	R.	DREW	MAJOR	

Residence Information (Select One) ◉ US Residency Non US Residency Active US Military Service

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
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.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

City	OREM	State/Province	UT	Country of Residence	US

Mailing Address of Inventor:

Address 1	863 WEST 600 SOUTH		
Address 2			
City	OREM	**State/Province**	UT
Postal Code	84058	**Country**	US

All Inventors Must Be Listed - Additional Inventor Information blocks may be generated within this form by selecting the **Add** button. Add

Correspondence Information:

Enter either Customer Number or complete the Correspondence Information section below.
For further information see 37 CFR 1.33(a).

☐ **An Address is being provided for the correspondence Information of this application.**

Customer Number	70560		
Email Address	docketing@lkglobal.com	Add Email	Remove Email

Application Information:

Title of the Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING		
Attorney Docket Number	066.0141X1C9	**Small Entity Status Claimed**	☐
Application Type	Nonprovisional		
Subject Matter	Utility		
Total Number of Drawing Sheets (if any)	11	**Suggested Figure for Publication (if any)**	

Filing By Reference:

Only complete this section when filing an application by reference under 35 U.S.C. 111(c) and 37 CFR 1.57(a). Do not complete this section if application papers including a specification and any drawings are being filed. Any domestic benefit or foreign priority information must be provided in the appropriate section(s) below (i.e., "Domestic Benefit/National Stage Information" and "Foreign Priority Information").

For the purposes of a filing date under 37 CFR 1.53(b), the description and any drawings of the present application are replaced by this reference to the previously filed application, subject to conditions and requirements of 37 CFR 1.57(a).

Application number of the previously filed application	Filing date (YYYY-MM-DD)	Intellectual Property Authority or Country

Publication Information:

☐ Request Early Publication (Fee required at time of Request 37 CFR 1.219)

☐ **Request Not to Publish.** I hereby request that the attached application not be published under 35 U.S.C. 122(b) and certify that the invention disclosed in the attached application **has not and will not** be the subject of an application filed in another country, or under a multilateral international agreement, that requires publication at eighteen months after filing.

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
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.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

Representative Information:

Representative information should be provided for all practitioners having a power of attorney in the application. Providing this information in the Application Data Sheet does not constitute a power of attorney in the application (see 37 CFR 1.32).
Either enter Customer Number or complete the Representative Name section below. If both sections are completed the customer Number will be used for the Representative Information during processing.

Please Select One:	● Customer Number	US Patent Practitioner	◯ Limited Recognition (37 CFR 11.9)
Customer Number	70560		

Domestic Benefit/National Stage Information:

This section allows for the applicant to either claim benefit under 35 U.S.C. 119(e), 120, 121, 365(c), or 386(c) or indicate National Stage entry from a PCT application. Providing benefit claim information in the Application Data Sheet constitutes the specific reference required by 35 U.S.C. 119(e) or 120, and 37 CFR 1.78.
When referring to the current application, please leave the "Application Number" field blank.

Prior Application Status	Pending				Remove
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)		
	Continuation of	16004056	2018-06-08		
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
16004056	Continuation of	15414025	2017-01-24	9998516	2018-06-12
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
15414025	Continuation of	14719122	2015-05-21	9571551	2017-02-14
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
14719122	Continuation of	14106051	2013-12-13	9071668	2015-06-30
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
14106051	Continuation of	13617114	2012-09-14	8612624	2013-12-17

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
13617114	Continuation of	12906940	2010-10-18	8402156	2013-03-19
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
12906940	Continuation of	11673483	2007-02-09	7818444	2010-10-19
Prior Application Status	Patented				Remove
Application Number	Continuity Type	Prior Application Number	Filing Date (YYYY-MM-DD)	Patent Number	Issue Date (YYYY-MM-DD)
11673483	Continuation in part of	11116783	2005-04-28	8868772	2014-10-21
Prior Application Status	Expired				Remove
Application Number	Continuity Type	Prior Application Number	Filing or 371(c) Date (YYYY-MM-DD)		
11116783	Claims benefit of provisional	60566831	2004-04-30		

Additional Domestic Benefit/National Stage Data may be generated within this form by selecting the **Add** button. Add

Foreign Priority Information:

This section allows for the applicant to claim priority to a foreign application. Providing this information in the application data sheet constitutes the claim for priority as required by 35 U.S.C. 119(b) and 37 CFR 1.55. When priority is claimed to a foreign application that is eligible for retrieval under the priority document exchange program (PDX)[i] the information will be used by the Office to automatically attempt retrieval pursuant to 37 CFR 1.55(i)(1) and (2). Under the PDX program, applicant bears the ultimate responsibility for ensuring that a copy of the foreign application is received by the Office from the participating foreign intellectual property office, or a certified copy of the foreign priority application is filed, within the time period specified in 37 CFR 1.55(g)(1).

Remove

Application Number	Country[i]	Filing Date (YYYY-MM-DD)	Access Code[i] (if applicable)

Additional Foreign Priority Data may be generated within this form by selecting the **Add** button. Add

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
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.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

☐ This application (1) claims priority to or the benefit of an application filed before March 16, 2013 and (2) also contains, or contained at any time, a claim to a claimed invention that has an effective filing date on or after March 16, 2013.
NOTE: By providing this statement under 37 CFR 1.55 or 1.78, this application, with a filing date on or after March 16, 2013, will be examined under the first inventor to file provisions of the AIA.

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
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.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

Authorization or Opt-Out of Authorization to Permit Access:

When this Application Data Sheet is properly signed and filed with the application, applicant has provided written authority to permit a participating foreign intellectual property (IP) office access to the instant application-as-filed (see paragraph A in subsection 1 below) and the European Patent Office (EPO) access to any search results from the instant application (see paragraph B in subsection 1 below).

Should applicant choose not to provide an authorization identified in subsection 1 below, applicant **must opt-out** of the authorization by checking the corresponding box A or B or both in subsection 2 below.

NOTE: This section of the Application Data Sheet is **ONLY** reviewed and processed with the **INITIAL** filing of an application. After the initial filing of an application, an Application Data Sheet cannot be used to provide or rescind authorization for access by a foreign IP office(s). Instead, Form PTO/SB/39 or PTO/SB/69 must be used as appropriate.

1. Authorization to Permit Access by a Foreign Intellectual Property Office(s)

A. Priority Document Exchange (PDX) - Unless box A in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the European Patent Office (EPO), the Japan Patent Office (JPO), the Korean Intellectual Property Office (KIPO), the State Intellectual Property Office of the People's Republic of China (SIPO), the World Intellectual Property Organization (WIPO), and any other foreign intellectual property office participating with the USPTO in a bilateral or multilateral priority document exchange agreement in which a foreign application claiming priority to the instant patent application is filed, access to: (1) the instant patent application-as-filed and its related bibliographic data, (2) any foreign or domestic application to which priority or benefit is claimed by the instant application and its related bibliographic data, and (3) the date of filing of this Authorization. See 37 CFR 1.14(h)(1).

B. Search Results from U.S. Application to EPO - Unless box B in subsection 2 (opt-out of authorization) is checked, the undersigned hereby **grants the USPTO authority** to provide the EPO access to the bibliographic data and search results from the instant patent application when a European patent application claiming priority to the instant patent application is filed. See 37 CFR 1.14(h)(2).

The applicant is reminded that the EPO's Rule 141(1) EPC (European Patent Convention) requires applicants to submit a copy of search results from the instant application without delay in a European patent application that claims priority to the instant application.

2. Opt-Out of Authorizations to Permit Access by a Foreign Intellectual Property Office(s)

☐ A. Applicant **DOES NOT** authorize the USPTO to permit a participating foreign IP office access to the instant application-as-filed. If this box is checked, the USPTO will not be providing a participating foreign IP office with any documents and information identified in subsection 1A above.

☐ B. Applicant **DOES NOT** authorize the USPTO to transmit to the EPO any search results from the instant patent application. If this box is checked, the USPTO will not be providing the EPO with search results from the instant application.

NOTE: Once the application has published or is otherwise publicly available, the USPTO may provide access to the application in accordance with 37 CFR 1.14.

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
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.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

Applicant Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

Applicant 1 Remove

If the applicant is the inventor (or the remaining joint inventor or inventors under 37 CFR 1.45), this section should not be completed. The information to be provided in this section is the name and address of the legal representative who is the applicant under 37 CFR 1.43; or the name and address of the assignee, person to whom the inventor is under an obligation to assign the invention, or person who otherwise shows sufficient proprietary interest in the matter who is the applicant under 37 CFR 1.46. If the applicant is an applicant under 37 CFR 1.46 (assignee, person to whom the inventor is obligated to assign, or person who otherwise shows sufficient proprietary interest) together with one or more joint inventors, then the joint inventor or inventors who are also the applicant should be identified in this section.

Clear

Assignee	Legal Representative under 35 U.S.C. 117	Joint Inventor
● Person to whom the inventor is obligated to assign.	Person who shows sufficient proprietary interest	

If applicant is the legal representative, indicate the authority to file the patent application, the inventor is:

Name of the Deceased or Legally Incapacitated Inventor:

If the Applicant is an Organization check here. ☒

Organization Name	DISH Technologies L.L.C.		
Mailing Address Information For Applicant:			
Address 1	9601 S. Meridian Blvd.		
Address 2			
City	Englewood	**State/Province**	CO
Country	US	Postal Code	80112
Phone Number		Fax Number	
Email Address			

Additional Applicant Data may be generated within this form by selecting the Add button. Add

Assignee Information including Non-Applicant Assignee Information:

Providing assignment information in this section does not substitute for compliance with any requirement of part 3 of Title 37 of CFR to have an assignment recorded by the Office.

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
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.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

Assignee 1

Complete this section if assignee information, including non-applicant assignee information, is desired to be included on the patent application publication. An assignee-applicant identified in the "Applicant Information" section will appear on the patent application publication as an applicant. For an assignee-applicant, complete this section only if identification as an assignee is also desired on the patent application publication.

Remove

If the Assignee or Non-Applicant Assignee is an Organization check here. ☒

Organization Name	DISH Technologies L.L.C.

Mailing Address Information For Assignee including Non-Applicant Assignee:

Address 1	9601 S. Meridian Blvd.		
Address 2			
City	Englewood	**State/Province**	CO
Country i	US	Postal Code	80112
Phone Number		Fax Number	
Email Address			

Additional Assignee or Non-Applicant Assignee Data may be generated within this form by selecting the Add button. Add

Signature:

Remove

NOTE: This Application Data Sheet must be signed in accordance with 37 CFR 1.33(b). **However, if this Application Data Sheet is submitted with the INITIAL filing of the application and either box A or B is not checked in subsection 2 of the "Authorization or Opt-Out of Authorization to Permit Access" section, then this form must also be signed in accordance with 37 CFR 1.14(c).**

This Application Data Sheet **must** be signed by a patent practitioner if one or more of the applicants is a **juristic entity** (e.g., corporation or association). If the applicant is two or more joint inventors, this form must be signed by a patent practitioner, **all** joint inventors who are the applicant, or one or more joint inventor-applicants who have been given power of attorney (e.g., see USPTO Form PTO/AIA/81) on behalf of **all** joint inventor-applicants.

See 37 CFR 1.4(d) for the manner of making signatures and certifications.

Signature	/BRETT A. CARLSON/			Date (YYYY-MM-DD)	2020-05-18
First Name	Brett A.	Last Name	Carlson	Registration Number	39,928

Additional Signature may be generated within this form by selecting the Add button. Add

PTO/AIA/14 (11-15)
Approved for use through 04/30/2017. OMB 0651-0032
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.

Application Data Sheet 37 CFR 1.76	Attorney Docket Number	066.0141X1C9
	Application Number	
Title of Invention	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING	

This collection of information is required by 37 CFR 1.76. 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 23 minutes to complete, including gathering, preparing, and submitting the completed application data sheet 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, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

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 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 CooperationTreaty.

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 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.

PATENT APPLICATION FEE DETERMINATION RECORD

Substitute for Form PTO-875

Application or Docket Number: 16/876,579

APPLICATION AS FILED - PART I

FOR	(Column 1) NUMBER FILED	(Column 2) NUMBER EXTRA	SMALL ENTITY RATE($)	SMALL ENTITY FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	OTHER THAN SMALL ENTITY FEE($)
BASIC FEE (37 CFR 1.16(a), (b), or (c))	N/A	N/A	N/A			N/A	300
SEARCH FEE (37 CFR 1.16(k), (i), or (m))	N/A	N/A	N/A			N/A	660
EXAMINATION FEE (37 CFR 1.16(o), (p), or (q))	N/A	N/A	N/A			N/A	760
TOTAL CLAIMS (37 CFR 1.16(i))	30 minus 20 =	* 10			OR	x 100 =	1000
INDEPENDENT CLAIMS (37 CFR 1.16(h))	4 minus 3 =	* 1				x 460 =	460
APPLICATION SIZE FEE (37 CFR 1.16(s))	If the specification and drawings exceed 100 sheets of paper, the application size fee due is $310 ($155 for small entity) for each additional 50 sheets or fraction thereof. See 35 U.S.C. 41(a)(1)(G) and 37 CFR 1.16(s).						0.00
MULTIPLE DEPENDENT CLAIM PRESENT (37 CFR 1.16(j))							0.00
* If the difference in column 1 is less than zero, enter "0" in column 2.			TOTAL			TOTAL	3180

APPLICATION AS AMENDED - PART II

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE($)	SMALL ENTITY ADDITIONAL FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE($)
AMENDMENT A	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

		(Column 1) CLAIMS REMAINING AFTER AMENDMENT		(Column 2) HIGHEST NUMBER PREVIOUSLY PAID FOR	(Column 3) PRESENT EXTRA	SMALL ENTITY RATE($)	SMALL ENTITY ADDITIONAL FEE($)	OR	OTHER THAN SMALL ENTITY RATE($)	OTHER THAN SMALL ENTITY ADDITIONAL FEE($)
AMENDMENT B	Total (37 CFR 1.16(i))	*	Minus	**	=	x =		OR	x =	
	Independent (37 CFR 1.16(h))	*	Minus	***	=	x =		OR	x =	
	Application Size Fee (37 CFR 1.16(s))									
	FIRST PRESENTATION OF MULTIPLE DEPENDENT CLAIM (37 CFR 1.16(j))							OR		
						TOTAL ADD'L FEE		OR	TOTAL ADD'L FEE	

* If the entry in column 1 is less than the entry in column 2, write "0" in column 3.
** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 20, enter "20".
*** If the "Highest Number Previously Paid For" IN THIS SPACE is less than 3, enter "3".
The "Highest Number Previously Paid For" (Total or Independent) is the highest found in the appropriate box in column 1.

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 NUMBER	FILING or 371(c) DATE	GRP ART UNIT	FIL FEE REC'D	ATTY.DOCKET.NO	TOT CLAIMS	IND CLAIMS
16/876,579	05/18/2020		3180	066.0141X1C9	30	4

CONFIRMATION NO. 3720

FILING RECEIPT

70560
LKGLOBAL (Dish Technologies L.L.C.)
7010 E. COCHISE ROAD
SCOTTSDALE, AZ 85253

OC000000117234022

Date Mailed: 05/27/2020

Receipt is acknowledged of this non-provisional utility patent application. The application will be taken up for examination in due course. Applicant will be notified as to the results of the examination. Any correspondence concerning the application must include the following identification information: the U.S. APPLICATION NUMBER, FILING DATE, NAME OF FIRST INVENTOR, and TITLE OF INVENTION. Fees transmitted by check or draft are subject to collection.

Please verify the accuracy of the data presented on this receipt. If an error is noted on this Filing Receipt, please submit a written request for a corrected Filing Receipt, including a properly marked-up ADS showing the changes with strike-through for deletions and underlining for additions. If you received a "Notice to File Missing Parts" or other Notice requiring a response for this application, please submit any request for correction to this Filing Receipt with your reply to the Notice. When the USPTO processes the reply to the Notice, the USPTO will generate another Filing Receipt incorporating the requested corrections provided that the request is grantable.

Inventor(s)

DAVID F. BRUECK, SARATOGA SPRINGS, UT;
MARK B. HURST, CEDAR HILLS, UT;
R. DREW MAJOR, OREM, UT;

Applicant(s)

DISH Technologies L.L.C., Englewood, CO

Assignment For Published Patent Application

DISH Technologies L.L.C., Englewood, CO

Power of Attorney: The patent practitioners associated with Customer Number 70560

Domestic Priority data as claimed by applicant

This application is a CON of 16/004,056 06/08/2018 PAT 10659513
which is a CON of 15/414,025 01/24/2017 PAT 9998516
which is a CON of 14/719,122 05/21/2015 PAT 9571551
which is a CON of 14/106,051 12/13/2013 PAT 9071668
which is a CON of 13/617,114 09/14/2012 PAT 8612624
which is a CON of 12/906,940 10/18/2010 PAT 8402156
which is a CON of 11/673,483 02/09/2007 PAT 7818444
which is a CIP of 11/116,783 04/28/2005 PAT 8868772
which claims benefit of 60/566,831 04/30/2004

Foreign Applications for which priority is claimed (You may be eligible to benefit from the **Patent Prosecution Highway** program at the USPTO. Please see http://www.uspto.gov for more information.) - None.

Foreign application information must be provided in an Application Data Sheet in order to constitute a claim to foreign priority. See 37 CFR 1.55 and 1.76.

Permission to Access Application via Priority Document Exchange: Yes

Permission to Access Search Results: Yes

Applicant may provide or rescind an authorization for access using Form PTO/SB/39 or Form PTO/SB/69 as appropriate.

If Required, Foreign Filing License Granted: 05/26/2020

The country code and number of your priority application, to be used for filing abroad under the Paris Convention, is **US 16/876,579**

Projected Publication Date: 09/03/2020

Non-Publication Request: No

Early Publication Request: No
Title

APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING

Preliminary Class

Statement under 37 CFR 1.55 or 1.78 for AIA (First Inventor to File) Transition Applications: No

PROTECTING YOUR INVENTION OUTSIDE THE UNITED STATES

Since the rights granted by a U.S. patent extend only throughout the territory of the United States and have no effect in a foreign country, an inventor who wishes patent protection in another country must apply for a patent in a specific country or in regional patent offices. Applicants may wish to consider the filing of an international application under the Patent Cooperation Treaty (PCT). An international (PCT) application generally has the same effect as a regular national patent application in each PCT-member country. The PCT process **simplifies** the filing of patent applications on the same invention in member countries, but **does not result** in a grant of "an international patent" and does not eliminate the need of applicants to file additional documents and fees in countries where patent protection is desired.

Almost every country has its own patent law, and a person desiring a patent in a particular country must make an application for patent in that country in accordance with its particular laws. Since the laws of many countries differ in various respects from the patent law of the United States, applicants are advised to seek guidance from specific foreign countries to ensure that patent rights are not lost prematurely.

Applicants also are advised that in the case of inventions made in the United States, the Director of the USPTO must issue a license before applicants can apply for a patent in a foreign country. The filing of a U.S. patent application serves as a request for a foreign filing license. The application's filing receipt contains further information and guidance as to the status of applicant's license for foreign filing.

Applicants may wish to consult the USPTO booklet, "General Information Concerning Patents" (specifically, the section entitled "Treaties and Foreign Patents") for more information on timeframes and deadlines for filing foreign

patent applications. The guide is available either by contacting the USPTO Contact Center at 800-786-9199, or it can be viewed on the USPTO website at http://www.uspto.gov/web/offices/pac/doc/general/index.html.

For information on preventing theft of your intellectual property (patents, trademarks and copyrights), you may wish to consult the U.S. Government website, http://www.stopfakes.gov. Part of a Department of Commerce initiative, this website includes self-help "toolkits" giving innovators guidance on how to protect intellectual property in specific countries such as China, Korea and Mexico. For questions regarding patent enforcement issues, applicants may call the U.S. Government hotline at 1-866-999-HALT (1-866-999-4258).

LICENSE FOR FOREIGN FILING UNDER

Title 35, United States Code, Section 184

Title 37, Code of Federal Regulations, 5.11 & 5.15

GRANTED

The applicant has been granted a license under 35 U.S.C. 184, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" followed by a date appears on this form. Such licenses are issued in all applications where the conditions for issuance of a license have been met, regardless of whether or not a license may be required as set forth in 37 CFR 5.15. The scope and limitations of this license are set forth in 37 CFR 5.15(a) unless an earlier license has been issued under 37 CFR 5.15(b). The license is subject to revocation upon written notification. The date indicated is the effective date of the license, unless an earlier license of similar scope has been granted under 37 CFR 5.13 or 5.14.

This license is to be retained by the licensee and may be used at any time on or after the effective date thereof unless it is revoked. This license is automatically transferred to any related applications(s) filed under 37 CFR 1.53(d). This license is not retroactive.

The grant of a license does not in any way lessen the responsibility of a licensee for the security of the subject matter as imposed by any Government contract or the provisions of existing laws relating to espionage and the national security or the export of technical data. Licensees should apprise themselves of current regulations especially with respect to certain countries, of other agencies, particularly the Office of Defense Trade Controls, Department of State (with respect to Arms, Munitions and Implements of War (22 CFR 121-128)); the Bureau of Industry and Security, Department of Commerce (15 CFR parts 730-774); the Office of Foreign AssetsControl, Department of Treasury (31 CFR Parts 500+) and the Department of Energy.

NOT GRANTED

No license under 35 U.S.C. 184 has been granted at this time, if the phrase "IF REQUIRED, FOREIGN FILING LICENSE GRANTED" DOES NOT appear on this form. Applicant may still petition for a license under 37 CFR 5.12, if a license is desired before the expiration of 6 months from the filing date of the application. If 6 months has lapsed from the filing date of this application and the licensee has not received any indication of a secrecy order under 35 U.S.C. 181, the licensee may foreign file the application pursuant to 37 CFR 5.15(b).

SelectUSA

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 U.S. offers tremendous resources and advantages for those who invest and manufacture goods here. Through SelectUSA, our nation works to promote and facilitate business investment. SelectUSA provides information assistance to the international investor community; serves as an ombudsman for existing and potential investors; advocates on behalf of U.S. cities, states, and regions competing for global investment; and counsels U.S. economic development organizations on investment attraction best practices. To learn more about why the United States is the best country in the world to develop technology, manufacture products, deliver services, and grow your business, visit http://www.SelectUSA.gov or call +1-202-482-6800.

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.
16/876,579	05/18/2020	DAVID F. BRUECK	066.0141X1C9	3720

70560 7590 06/03/2020

LKGLOBAL (Dish Technologies L.L.C.)
7010 E. COCHISE ROAD
SCOTTSDALE, AZ 85253

EXAMINER
PATEL, CHIRAG R

ART UNIT	PAPER NUMBER
2454	

NOTIFICATION DATE	DELIVERY MODE
06/03/2020	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):

dish@lkglobal.com
ipdept@dish.com

Decision Granting Request for Prioritized Examination (Track I)	**Application No.** 16/876,579	**Applicant(s)** BRUECK et al.	
	Examiner CHERYL P GIBSON BAYLOR	**Art Unit** OPET	**AIA (FITF) Status** No

1. THE REQUEST FILED 18 May 2020 IS **GRANTED** .

The above-identified application has met the requirements for prioritized examination

A. ☑ for an original nonprovisional application (Track I).
B. ☐ for an application undergoing continued examination (RCE).

2. **The above-identified application will undergo prioritized examination**. The application will be accorded special status throughout its entire course of prosecution until one of the following occurs:

A. filing a **petition for extension of time** to extend the time period for filing a reply;

B. filing an **amendment to amend the application to contain more than four independent claims, more than thirty total claims** , or a multiple dependent claim;

C. filing a **request for continued examination** ;

D. filing a notice of appeal;

E. filing a request for suspension of action;

F. mailing of a notice of allowance;

G. mailing of a final Office action;

H. completion of examination as defined in 37 CFR 41.102; or

I. abandonment of the application.

Telephone inquiries with regard to this decision should be directed to CHERYL GIBSON BAYLOR at (571)272-3213. In his/her absence, calls may be directed to Petition Help Desk at (571) 272-3282.

/CHERYL GIBSON BAYLOR/ Paralegal Specialist, OPET	

Bibliographic Data

Application No: 16/876,579

Foreign Priority claimed: ○ Yes ◉ No

35 USC 119 (a-d) conditions met: ☐ Yes ☑ No ☐ Met After Allowance

Verified and Acknowledged: /CHIRAG R PATEL/ Examiner's Signature | Initials

Title: APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING

FILING or 371(c) DATE	CLASS	GROUP ART UNIT	ATTORNEY DOCKET NO.
05/18/2020	709	2454	066.0141X1C9
RULE			

APPLICANTS

DISH Technologies L.L.C., Englewood, CO, UNITED STATES

INVENTORS

DAVID F. BRUECK SARATOGA SPRINGS, UT, UNITED STATES

MARK B. HURST CEDAR HILLS, UT, UNITED STATES

R. DREW MAJOR OREM, UT, UNITED STATES

CONTINUING DATA

This application is a CON of 16004056 06/08/2018 PAT 10659513

16004056 is a CON of 15414025 01/24/2017 PAT 9998516

15414025 is a CON of 14719122 05/21/2015 PAT 9571551

14719122 is a CON of 14106051 12/13/2013 PAT 9071668

14106051 is a CON of 13617114 09/14/2012 PAT 8612624

13617114 is a CON of 12906940 10/18/2010 PAT 8402156

12906940 is a CON of 11673483 02/09/2007 PAT 7818444

11673483 is a CIP of 11116783 04/28/2005 PAT 8868772

11116783 has PRO of 60566831 04/30/2004

FOREIGN APPLICATIONS

IF REQUIRED, FOREIGN LICENSE GRANTED**

05/26/2020

STATE OR COUNTRY

UNITED STATES

ADDRESS

LKGLOBAL (Dish Technologies L.L.C.)

7010 E. COCHISE ROAD
SCOTTSDALE, AZ 85253
UNITED STATES

FILING FEE RECEIVED

$7,320

Notice of References Cited	Application/Control No. 16/876,579	Applicant(s)/Patent Under Reexamination BRUECK et al.	
	Examiner CHIRAG R PATEL	Art Unit 2454	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	CPC Classification	US Classification
*	A	US-7369610-B2	05-2008	Xu; Jizheng	H04N21/2662	375/240.08
*	B	US-20050055425-A1	03-2005	Lango, Jason	H04N7/17318	709/219
*	C	US-20040143672-A1	07-2004	Padmanabham, Venkata N.	H04L65/4084	709/231
	D					
	E					
	F					
	G					
	H					
	I					
	J					
	K					
	L					
	M					

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	CPC Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Search Notes	Application/Control No.	Applicant(s)/Patent Under Reexamination
	16/876,579	BRUECK et al.
	Examiner	Art Unit
	CHIRAG R PATEL	2454

CPC - Searched*		
Symbol	Date	Examiner
(H04N21/2662, H04N21/8456, H04L65/607,H04L65/608, H04L67/02	06/12/2020	C.P.

CPC Combination Sets - Searched*		
Symbol	Date	Examiner

US Classification - Searched*			
Class	Subclass	Date	Examiner

* See search history printout included with this form or the SEARCH NOTES box below to determine the scope of the search.

Search Notes		
Search Notes	Date	Examiner
EAST	06/12/2020	C.P.

Interference Search			
US Class/CPC Symbol	US Subclass/CPC Group	Date	Examiner

Aylo Exhibit 1002
Page 84 of 1405

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.
16/876,579	05/18/2020	DAVID F. BRUECK	066.0141X1C9	3720

70560 7590 06/18/2020

LKGLOBAL (Dish Technologies L.L.C.)
7010 E. COCHISE ROAD
SCOTTSDALE, AZ 85253

EXAMINER
PATEL, CHIRAG R

ART UNIT	PAPER NUMBER
2454	

NOTIFICATION DATE	DELIVERY MODE
06/18/2020	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):

dish@lkglobal.com
ipdept@dish.com

Office Action Summary	Application No. 16/876,579	Applicant(s) BRUECK et al.	
	Examiner CHIRAG R PATEL	Art Unit 2454	AIA (FITF) Status No

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

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTHS FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☑ Responsive to communication(s) filed on the filing on May 18, 2020.

☐ A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on _____.

2a)☐ This action is **FINAL.** 2b) ☑ This action is non-final.

3)☐ 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.

4)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims*

5) ☑ Claim(s) 1-30 is/are pending in the application.

5a) Of the above claim(s) _____ is/are withdrawn from consideration.

6) ☐ Claim(s) _____ is/are allowed.

7) ☑ Claim(s) 1-30 is/are rejected.

8) ☐ Claim(s) _____ is/are objected to.

9) ☐ Claim(s) _____ are subject to restriction and/or election requirement

* If any claims have been determined allowable, 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.**

Application Papers

10)☐ The specification is objected to by the Examiner.

11)☑ The drawing(s) filed on 18 May 2020 is/are: a)☑ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

Priority under 35 U.S.C. § 119

12)☐ 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)).

** See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☑ Notice of References Cited (PTO-892)

2) ☐ Information Disclosure Statement(s) (PTO/SB/08a and/or PTO/SB/08b) Paper No(s)/Mail Date _____.

3) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date _____.

4) ☐ Other: _____.

Notice of Pre-AIA or AIA Status

The present application is being examined under the pre-AIA first to invent provisions.

Double Patenting

A rejection based on double patenting of the "same invention" type finds its support in the language of 35 U.S.C. 101 which states that "whoever invents or discovers any new and useful process... may obtain a patent therefor..." (Emphasis added). Thus, the term "same invention," in this context, means an invention drawn to identical subject matter. See *Miller v. Eagle Mfg. Co.*, 151 U.S. 186 (1894); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA 1970); *In re Ockert*, 245 F.2d 467, 114 USPQ 330 (CCPA 1957).

A statutory type (35 U.S.C. 101) double patenting rejection can be overcome by canceling or amending the claims that are directed to the same invention so they are no longer coextensive in scope. The filing of a terminal disclaimer cannot overcome a double patenting rejection based upon 35 U.S.C. 101.

Claims 1-2, 4-21, 23-26, 28 and 30 are rejected under 35 U.S.C. 101 as claiming the same invention as that of claims 1-5, 7, 9-13, 16-19 and 23-28 of U.S. Patent No. 10,469,554. This is a statutory double patenting rejection.

Claims 1 and 2 of the instant application are a verbatim of claim 1 of the patent.

Claim 4 of the instant application is a verbatim of claim 2 of the patent.

Claim 5 of the instant application is a verbatim of claim 3 of the patent.

Claim 6 of the instant application is a verbatim of claim 4 of the patent.

Claim 7 of the instant application is a verbatim of claim 5 of the patent.

Claim 8 of the instant application is a verbatim of claim 7 of the patent.

Claim 11 of the instant application is a verbatim of claim 9 of the patent.

Claim 12 of the instant application is a verbatim of claim 10 of the patent.

Claim 13 of the instant application is a verbatim of claim 11 of the patent.

Claim 14 of the instant application is a verbatim of claim 12 of the patent.

Claim 15 of the instant application is a verbatim of claim 13 of the patent.

Claims 14 and 21 of the instant application are a verbatim of claim 16 of the patent.

Claim 15 of the instant application is a verbatim of claim 17 of the patent.

Claim 16 of the instant application is a verbatim of claim 18 of the patent.

Claim 17 of the instant application is a verbatim of claim 19 of the patent.

Claim 18 of the instant application is a verbatim of claim 23 of the patent.

Claim 19 of the instant application is a verbatim of claim 24 of the patent.

Claim 20 of the instant application is a verbatim of claim 25 of the patent.

Claims 23 and 26 of the instant application are a verbatim of claim 26 of the patent.

Claim 24 of the instant application is a verbatim of claim 27 of the patent.

Claim 25 of the instant application is a verbatim of claim 28 of the patent.

Claim 23 of the instant application is a verbatim of claim 26 of the patent.

Claims 28 and 30 of the instant application are a verbatim of claim 26 of the patent.

The nonstatutory double patenting rejection is based on a judicially created doctrine grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or improper timewise extension of the “right to exclude” granted by a patent and to prevent possible harassment by multiple assignees. A nonstatutory double patenting rejection is appropriate where the conflicting claims are not identical, but at least one examined application claim is not patentably distinct from the reference claim(s) because the examined application claim is either anticipated by, or would have been obvious over, the reference claim(s). See, e.g., *In re Berg*, 140 F.3d 1428, 46 USPQ2d 1226 (Fed. Cir. 1998); *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed. Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686 F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d

438, 164 USPQ 619 (CCPA 1970); *In re Thorington*, 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) or 1.321(d) may be used to overcome an actual or provisional rejection based on nonstatutory double patenting provided the reference application or patent either is shown to be commonly owned with the examined application, or claims an invention made as a result of activities undertaken within the scope of a joint research agreement. See MPEP § 717.02 for applications subject to examination under the first inventor to file provisions of the AIA as explained in MPEP § 2159. See MPEP §§ 706.02(l)(1) - 706.02(l)(3) for applications not subject to examination under the first inventor to file provisions of the AIA. A terminal disclaimer must be signed in compliance with 37 CFR 1.321(b).

The USPTO Internet website contains terminal disclaimer forms which may be used. Please visit www.uspto.gov/patent/patents-forms. The filing date of the application in which the form is filed determines what form (e.g., PTO/SB/25, PTO/SB/26, PTO/AIA/25, or PTO/AIA/26) should be used. A web-based eTerminal Disclaimer may be filled out completely online using web-screens. An eTerminal Disclaimer that meets all requirements is auto-processed and approved immediately upon submission. For more information about eTerminal Disclaimers, refer to www.uspto.gov/patents/process/file/efs/guidance/eTD-info-I.jsp.

Claims 1-3, 6-8, 14, 17, 21-23, and 26-30 are rejected on the ground of nonstatutory double patenting as being unpatentable over claims 1, 4, 5 and 7 of U.S. Patent No. 10,469,555. Although the claims at issue are not identical, they are not patentably distinct from each other

Claims 1-3 of the instant application is broader and obvious over claim 1 of the patent. It would have been obvious for duration of the streamlet of the patent to be a same portion of the

video of the instant application. It would have been obvious for the group of streamlets of the patent to include a first and second streamlet of the instant application which corresponds to the portion of the video. Other differences it would have been obvious for the one storage device storing live event video to be an archived content of the instant application.

Claim 6 of the instant application is anticipated by claim 4 of the patent.

Claim 7 of the instant application is anticipated by claim 5 of the patent.

Claim 8 of the instant application is anticipated by claim 7 of the patent.

Claims 14 and 21-22 of the instant application is broader and obvious over claim 1 of the patent similar to reasons above as for claims 1-3.

Claim 17 of the instant application is anticipated by claim 5 of the patent.

Claims 23 and 26-27 of the instant application is anticipated by claims 1 and 4 of the patent similar to reasons above as for claims 1-3

Claims 28-30 of the instant application is anticipated by claims 1 and 4 of the patent similar to reasons above as for claims 1-3

Conclusion

The prior art made of record and not relied upon is considered pertinent to applicant's disclosure.

Xu et al. (US 7,369,610) - Enhancement layer switching for scalable video coding.

Lango et al. – (US 2005/0055425) - Pre-computing streaming media payload method and apparatus.

Padmanabham et al. (US 2004/0143672) - System and method for distributing streaming content through cooperative networking.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Chirag R Patel whose telephone number is (571)272-7966. The examiner can normally be reached on Monday to Friday from 8:00AM to 4:30PM. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Glenton Burgess, can be reached on 571-272-3949. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pairdirect.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll free).

/Chirag R Patel/
Primary Examiner, Art Unit 2454

Index of Claims	Application/Control No.	Applicant(s)/Patent Under Reexamination
	16/876,579	BRUECK et al.
	Examiner	Art Unit
	CHIRAG R PATEL	2454

✓	Rejected	-	Cancelled	N	Non-Elected	A	Appeal
=	Allowed	÷	Restricted	I	Interference	O	Objected

CLAIMS

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

CLAIM		DATE								
Final	Original	06/12/2020								
	1	✓								
	2	✓								
	3	✓								
	4	✓								
	5	✓								
	6	✓								
	7	✓								
	8	✓								
	9	✓								
	10	✓								
	11	✓								
	12	✓								
	13	✓								
	14	✓								
	15	✓								
	16	✓								
	17	✓								
	18	✓								
	19	✓								
	20	✓								
	21	✓								
	22	✓								
	23	✓								
	24	✓								
	25	✓								
	26	✓								
	27	✓								
	28	✓								
	29	✓								
	30	✓								

EAST Search History

EAST Search History (Prior Art)

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
S1	1	("10659513").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 07:55
S2	8	(("9571551") or ("9071668") or ("8612624") or ("8402156") or ("7818444") or ("8868772") or ("10469554") or ("10659513")).PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 07:58
S3	2	(("73369610") or ("20050055425") or ("20030103571")).PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 10:29
S4	3	(("7369610") or ("20050055425") or ("20030103571")).PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 10:30
S5	27	encod$3 near2 streamlets with bitrates	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 10:36
S6	1	("9751551").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 10:45
S7	1	("9571551").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 10:46
S8	1	("8402156").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 10:53
S9	1	("7818444").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 10:55
S10	7	(("20080162713") or ("20040170392") or ("20040143672") or ("6486803") or ("20030204519") or ("6486803") or ("20030005455") or ("20040054551") or ("20040170392")).PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 11:02
S11	54	streamlets with bitrates	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 11:33
S12	74	((H04N21/2662 OR H04N21/8456 OR H04L65/607 OR H04L65/608 OR	US-PGPUB; USPAT;	OR	ON	2020/06/12 12:37

		H04L67/02).CPC.) and streamlet near3 encod$3	USOCR; FPRS; EPO			
S13	0	((H04N21/2662 OR H04N21/8456 OR H04L65/607 OR H04L65/608 OR H04L67/02).CPC.) and streamlet near3 encod$3 and @rlad<"20040430"	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 13:03
S14	5	((H04N21/2662 OR H04N21/8456 OR H04L65/607 OR H04L65/608 OR H04L67/02).CPC.) and streamlets and @rlad<"20040430"	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 13:03
S15	33,997	portions with stream and @rlad<"20040430"	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 13:28
S16	1,761,238	portions with stream with bitrat$3and @rlad<"20040430"	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 13:29
S17	24	portions with stream with bitrat$3 and @rlad<"20040430"	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 13:29
S18	2	((H04N21/2662 OR H04N21/8456 OR H04L65/607 OR H04L65/608 OR H04L67/02).CPC.) and divid$3 with stream same bitrate and @rlad<"20040430"	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 13:30
S19	3	((H04N21/2662 OR H04N21/8456 OR H04L65/607 OR H04L65/608 OR H04L67/02).CPC.) and (divid$3 split$3)with stream same bitrate and @rlad<"20040430"	US-PGPUB; USPAT; USOCR; FPRS; EPO	OR	ON	2020/06/12 13:30
S20	1	("10469554").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 14:26
S21	1	("10469554").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 14:44
S22	1	("10469555").PN.	US-PGPUB; USPAT; USOCR	OR	OFF	2020/06/12 14:47

EAST Search History (Interference)

<This search history is empty>

6/12/2020 5:39:27 PM
C:\Users\cpatel\Documents\EAST\Workspaces\16876579.wsp

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 NUMBER	FILING OR 371(C) DATE	FIRST NAMED APPLICANT	ATTY. DOCKET NO./TITLE
16/876,579	05/18/2020	DAVID F. BRUECK	066.0141X1C9

CONFIRMATION NO. 3720

70560
LKGLOBAL (Dish Technologies L.L.C.)
7010 E. COCHISE ROAD
SCOTTSDALE, AZ 85253

PUBLICATION NOTICE

OC00000019661767

Title:APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING

Publication No.US-2020-0280594-A1
Publication Date:09/03/2020

NOTICE OF PUBLICATION OF APPLICATION

The above-identified application will be electronically published as a patent application publication pursuant to 37 CFR 1.211, et seq. The patent application publication number and publication date are set forth above.

The publication may be accessed through the USPTO's publically available Searchable Databases via the Internet at www.uspto.gov. The direct link to access the publication is currently http://www.uspto.gov/patft/.

The publication process established by the Office does not provide for mailing a copy of the publication to applicant. A copy of the publication may be obtained from the Office upon payment of the appropriate fee set forth in 37 CFR 1.19(a)(1). Orders for copies of patent application publications are handled by the USPTO's Public Records Division. The Public Records Division can be reached by telephone at (571) 272-3150 or (800) 972-6382, by facsimile at (571) 273-3250, by mail addressed to the United States Patent and Trademark Office, Public Records Division, Alexandria, VA 22313-1450 or via the Internet.

In addition, information on the status of the application, including the mailing date of Office actions and the dates of receipt of correspondence filed in the Office, may also be accessed via the Internet through the Patent Electronic Business Center at www.uspto.gov using the public side of the Patent Application Information and Retrieval (PAIR) system. The direct link to access this status information is currently https://portal.uspto.gov/pair/PublicPair. Prior to publication, such status information is confidential and may only be obtained by applicant using the private side of PAIR.

Further assistance in electronically accessing the publication, or about PAIR, is available by calling the Patent Electronic Business Center at 1-866-217-9197.

Office of Data Managment, Application Assistance Unit (571) 272-4000, or (571) 272-4200, or 1-888-786-0101

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
26 January 2006 (26.01.2006)

PCT

(10) International Publication Number
WO 2006/010113 A2

(51) International Patent Classification:
G06F 15/173 (2006.01)

(21) International Application Number:
PCT/US2005/024517

(22) International Filing Date: 11 July 2005 (11.07.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/586,952 9 July 2004 (09.07.2004) US

(71) Applicant *(for all designated States except US)*: **NETWORK FOUNDATION TECHNOLOGIES, LLC** [US/US]; 2215 Cooktown Road, Ruston, LA 71270 (US).

(72) Inventors: **O'NEAL, Mike**; 2215 Cooktown Road, Ruston, LA 71270 (US). **TALTON, John P.**; 2215 Cooktown Road, Ruston, LA 71270 (US).

(74) Agents: **LANDA, Adam B.** et al.; Greenberg Traurig, MetLife Building, 200 Park Avenue, New York, NY 10166 (US).

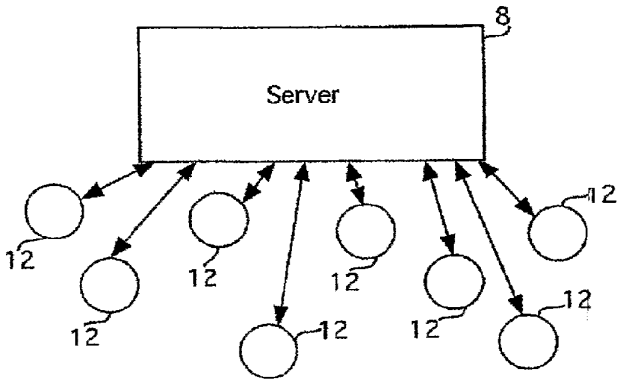
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *without international search report and to be republished upon receipt of that report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: SYSTEMS FOR DISTRIBUTING DATA OVER A COMPUTER NETWORK AND METHODS FOR ARRANGING NODES FOR DISTRIBUTION OF DATA OVER A COMPUTER NETWORK



PRIOR ART

(57) Abstract: Various embodiments of the present invention relate to a system for distributing data (e.g., content data) over a computer network and a method of arranging receiver nodes in a computer network such that the capacity of a server is effectively increased (e.g., the capacity of a server may be effectively multiplied many times over; the capacity of the server may be effectively increased exponentially). In one embodiment the present invention may take advantage of the excess capacity many receiver nodes possess, and may use such receiver nodes as repeaters. The distribution system may include node(s) having database(s) which indicate ancestor(s) and/or descendant(s) of the node so that reconfiguration of the distribution network may be accomplished without burdening the system's primary server. An embodiment of the present invention may include a process for configuring a computer information distribution network having a primary server node and user nodes docked in a cascaded relationship, and reconfiguring the network in the event that a user node departs from the network. In one example (which example is intended to be illustrative and not restrictive), the process may include the steps of providing a new user node (or connection requesting user node) with a connection address list of nodes within the network, having the new user node (or connection requesting user node) go to (or attempt to go to) the node at the top of the connection address list, determine whether that node is still part of the distribution network, and connect thereto if it is, and if it is not, to go to (or attempt to go to) the next node on the connection address list. In another example (which example is intended to be illustrative and not restrictive), when a user node departs from the distribution network, a propagation signal may be transmitted to the nodes below it in the network, causing them to move up in the network in a predetermined order. In another example (which example is intended to be illustrative and not restrictive), the present invention may provide a decentralized approach which provides, to each new user node (or connection requesting user node) a path back to the root server.

WO 2006/010113 A2

SYSTEMS FOR DISTRIBUTING DATA OVER A COMPUTER NETWORK AND METHODS FOR ARRANGING NODES FOR DISTRIBUTION OF DATA OVER A COMPUTER NETWORK

FIELD OF THE INVENTION

This patent application is related to and claims the benefit of the filing date of U.S. Provisional Patent Application Serial No. 60/586,952 for "Systems for Distributing Data Over A Computer Network And Methods For Arranging Nodes For Distribution Of Data Over A Computer Network", filed July 9, 2004.

Various embodiments of the present invention relate to systems for distributing data (e.g., content data) over a computer network and methods of arranging nodes for distribution of data (e.g., content data) over a computer network. In one example (which example is intended to be illustrative and not restrictive), the systems and methods of the present invention may be applied to the distribution of streaming audiovisual data over the Internet.

For the purposes of the present application the term "node" (e.g., as used in the phrase a first node communicates with a second node) is intended to refer to a computer system (or sometimes just "computer") configured to send and/or receive data over a computer network (e.g., a computer network including the Internet, a local-area-network, a wide-area-network, a wireless network). Since a "node" in a computer network would not exist but for a computer system being available on such node, the terms "node" and "computer system" (or sometimes just "computer") may be used interchangeably (i.e., the term "node" should be understood to include an available computer system (or sometime just "computer").

Further, for the purposes of the present application the term "upstream" (e.g., as used in the phrase a first computer system sends data via an upstream connection to a second computer system) is intended to refer to the communication path between a first computer system and the Internet when the first computer system is sending data to a second computer system via the Internet.

Further still, for the purposes of the present application the term "downstream" (e.g., as used in the phrase a first computer system receives data via a downstream connection from a second computer system) is intended to refer to the communication path between a first computer system and the Internet when the first computer system is receiving data from a second computer system via the Internet.

Further still, for the purposes of the present application the term "uptree" (e.g., as

used in the phrase a first node sends data via an uptree connection to a second node) is intended to refer to the network topology communication path between a first node and a second node when a first node is sending data to a second node which is higher up on the network topology tree (that is, closer to the root server).

Further still, for the purposes of the present application the term "downtree" (e.g., as used in the phrase a first node sends data via a downtree connection to a second node) is intended to refer to the network topology communication path between a first node and a second node when a first node is sending data to a second node which is lower down on the network topology tree (that is, closer to the leaves).

Further still, for the purposes of the present application the term "docked" (e.g., as used in the phrase a child node docks with a parent node) is intended to refer to forming a connection between nodes via which data may flow in at least one direction (e.g., at least uptree or downtree).

Further still, for the purposes of the present application the term "apparent available capacity" (e.g., as used in the phrase apparent available capacity to transmit content data) is intended to refer to a nominal capacity (e.g., a node's presumed capacity based upon a design specification but not necessarily actually tested in the network).

BACKGROUND OF THE INVENTION

In a computer network such as the Internet, each node in the network has an address. A computer system resident at a particular address may have sufficient bandwidth or capacity to receive data from, and to transmit data to, many other computer systems at other addresses. An example of such a computer system is a server, many commercial versions of which can simultaneously exchange data with thousands of other computer systems.

A computer system at another location may have only sufficient bandwidth to effectively exchange data with only one other computer system. An example of such a system is an end user's personal computer connected to the Internet by a very low speed dialup modem. However, even typical personal computers connected to the Internet by higher speed dialup modems may have sufficient bandwidth that they may exchange data essentially simultaneously in an effective manner with several other computer systems. Moreover, an end user's personal computer system may have even greater bandwidth when connected to the Internet by ISDN lines, DSL (e.g., ADSL) lines, cable modems, T1 lines or even higher capacity links. As discussed more fully below, various embodiments of the present invention

may take advantage of the availability of such higher capacity end user systems (e.g., those computer systems capable of essentially simultaneously exchanging data with multiple computer systems).

In a typical situation, as shown in FIG. 1, a content provider distributes its data by making the data available on a server node 8 simultaneously to a plurality of users at user nodes 12 (of note, the terms "server", "root server" and "primary server" may be used interchangeably throughout the present application to refer to the same device (i.e., the highest level parent node in a given network). The double-headed arrows show the two-way communication between each end user's computer system and the server. Essentially the content provider's server transmits a separate stream of signals to each receiver node. To accommodate additional users, the content provider would typically either add equipment to increase capacity or it would engage a mirror site to accomplish essentially the same result as adding equipment. The capacities of the end user computers is of virtually no consequence in such a system.

Another system for distributing data is the Napster™ music file exchange system provided by Napster, Inc. of Redwood City, Calif. A schematic of the Napster™ music file exchange system (as its operation is presently understood) is illustrated in FIG. 2.

More particularly, it is believed that in the Napster™ system a copy of the music data is not kept on the server. The server 9 instead maintains a database relating to the various music files on the computers of users who are logged onto the server 9. When a first user 12a sees that a desired music file is available from a second logged on user 12b, the first user causes his computer to query the server 9 for the second user's node address and a connection is made between the first and second user's computers through which the first user's computer notifies the second user's computer of the desired file and the second user's computer responds by transmitting a copy of the desired music file directly to the first user's computer. It is further believed that a first user attempting to download a particular file from a second user must start completely over again if the second user cancels its transmission or goes off line during the data transfer.

In another area, engineers have developed what is known as "streaming media." In summary, streaming media is a series of packets (e.g., of compressed data), each packet representing moving images and/or audio.

To help understand streaming media in more detail, it is helpful to review the traditional Internet distribution method. Each node (whether it is a server node or a user node) in a computer network has a unique identification (sometimes referred to as an "IP" address)

associated with it. On the Internet, the unique address may be referred to as a Uniform Resource Locator ("URL"). A user desiring to obtain data from a particular server enters that server's URL into the user's browser program. The browser program causes a connection request signal to be sent over the Internet to the server. If the server has the capacity to accept the connection, the connection is made between the server and the user node (files requested by the user are typically transmitted by the server in full to the user node and the browser program may store the files in buffer memory and display the content on the user's computer system monitor - some files may be more permanently stored in the computer system's memory for later viewing or playing.) The connection with the server is typically terminated once the files have been received at the user node (or the connection may be terminated a short time thereafter). Either way, the connection is usually of a very short time duration.

With streaming media, the contact between the server and user nodes is essentially continuous. When a connection between a server node and user node is made and streaming media is requested, the server sends streaming media packets of data to the user node. A streaming media player installed on the user's computer system (e.g., software, such as RealMedia™ from RealNetworks, Inc. of Seattle, Wash.,) causes the data to be stored in buffer memory. The player decompresses the data and begins playing the moving images and/or audio represented by the streaming media data on the user's computer system. As the data from a packet is played, the buffer containing that packet is emptied and becomes available to receive a new packet of data. As a result, the memory assets of a user's computer are not overly taxed. Continuous action content, such as, for example, the display of recorded motion picture films, videos or television shows may be distributed and played in essentially "real time," and live events, such as, for example, concerts, football games, court trials, and political debates may be transmitted and viewed essentially "live" (with only the brief delays needed for compression of the data being made available on the server, transmission from the server to the user node, and decompression and play on the user's computer system preventing a user from seeing the event at the exact same moment in time as a person actually at the event). And, when the systems are working as designed, the server node and user node may stay connected to each other until all the packets of data representing the content have been transmitted.

SUMMARY OF THE INVENTION

Various embodiments of the present invention relate to a system for distributing data (e.g., content data) over a computer network and a method of arranging receiver nodes in a computer network such that the capacity of a server is effectively increased (e.g., the capacity of a server may be effectively multiplied many times over; the capacity of the server may be effectively increased exponentially). In one embodiment the present invention may take advantage of the excess capacity many receiver nodes possess, and may use such receiver nodes as repeaters. The distribution system may include node(s) having database(s) which indicate ancestor(s) and/or descendant(s) of the node so that reconfiguration of the distribution network may be accomplished without burdening the system's primary server. An embodiment of the present invention may include a process for configuring a computer information distribution network having a primary server node and user nodes docked (i.e., connected) in a cascaded relationship, and reconfiguring the network in the event that a user node departs from the network. In one example (which example is intended to be illustrative and not restrictive), the process may include the steps of providing a new user node (or connection requesting user node) with a connection address list of nodes within the network, having the new user node (or connection requesting user node) go to (or attempt to go to) the node at the top of the connection address list, determine whether that node is still part of the distribution network, and connect thereto if it is, and if it is not, to go to (or attempt to go to) the next node on the connection address list. In another example (which example is intended to be illustrative and not restrictive), when a user node departs from the distribution network, a propagation signal may be transmitted to the nodes below it in the network, causing them to move up in the network in a predetermined order. In another example (which example is intended to be illustrative and not restrictive), the present invention may provide a decentralized approach which provides, to each new user node (or connection requesting user node) a path back to the root server.

Of note, various embodiments of the present invention may be applied to the transmission (e.g., the "appointment" transmission) of audiovisual content such as, for example (which example is intended to be illustrative and not restrictive), live or pre-recorded concerts, football games, court trials, political debates, motion picture films, videos or television shows. Such transmission of audiovisual content may be made with acceptable levels of quality, wherein "acceptable levels of quality" may vary depending upon the end users and the type of transmission. Further, such transmission of audiovisual content may be

carried out using streaming media transmissions intended to reach large audiences, in much the way that television shows transmitted over television cable and broadcast media reach large audiences (in this regard, the present invention may enable a server to transmit streaming media to the large number of users which would be essentially simultaneously logging on to view a particular audiovisual presentation under this television-type "large audience" transmission).

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a schematic drawing of a prior art computer information distribution network;

FIG. 2 is a schematic drawing of another prior art computer information distribution network;

FIG. 3 is a schematic drawing of an embodiment of a computer information distribution network formed pursuant to the present invention;

FIG. 4 is a schematic drawing of another embodiment of a computer information distribution network formed pursuant to the present invention;

FIG. 5 is a schematic drawing of another embodiment of a computer information distribution network formed pursuant to the present invention;

FIG. 6 is a schematic drawing of a particular topology of a computer information distribution network formed pursuant to an embodiment of the present invention;

FIG. 7 is a schematic drawing of a particular topology of the computer information distribution network formed pursuant to an embodiment of the present invention as shown in FIG. 6 (after the occurrence of an event);

FIG. 8 is a schematic drawing of another topology of a computer information distribution network formed pursuant to an embodiment of the present invention;

FIG. 9 is a schematic drawing of the topology of the computer information distribution network formed pursuant to an embodiment of the present invention as shown in FIG. 8 (after the occurrence of an event);

FIG. 10 is a flow diagram of an embodiment of the present invention showing a Server Connection Routine which may be performed when a prospective child node seeks to join the distribution network;

FIG. 11 is a flow diagram of an embodiment of the present invention showing a Prospective Child Node's Request to Server for a Connection Routine;

FIGS. 12A-12E are schematic drawings showing varying topologies of the computer information distribution network formed pursuant to an embodiment of the present invention under several circumstances;

FIG. 13 is a block diagram of an embodiment of the present invention showing the memory blocks into which the software used may partition a user node's memory;

FIG. 14 is a flow diagram of an embodiment of the present invention showing a Prospective Child Node's Connection Routine - the routine which a new user node (or connection requesting user node) may go through in attempting to connect to a distribution chain (or tree) after receiving a connection address list;

FIG. 15 is a flow diagram of an embodiment of the present invention showing a Prospective Child Node's Connection Routine With Return to Server Subroutine;

FIG. 16 is a flow diagram of an embodiment of the present invention illustrating a Prospective Parent Node's Connection Routine;

FIG. 17 is a flow diagram of an embodiment of the present invention illustrating a Server's Connection Instruction Routine;

FIG. 18 is a flow diagram of an embodiment of the present invention illustrating a Fully Occupied Parent's Connection Instruction Routine;

FIG. 19 is a flow diagram of an embodiment of the present invention illustrating a Multi-Node Selection Subroutine;

FIGS. 20A and 20B are together a flow diagram of an embodiment of the present invention illustrating a Universal Connection Address Routine;

FIG. 21 is a schematic drawing of a topology of a computer information distribution network of an embodiment of the present invention before a new node will be added using the Universal Connection Address Routine;

FIG. 22 is a schematic drawing of a topology of the computer information distribution network of an embodiment of the present invention when a new node is added using the Universal Connection Address Routine;

FIG. 23 is a schematic drawing of a topology of a computer information distribution network of an embodiment of the present invention before a reconfiguration event;

FIG. 24 is a schematic drawing of a topology of a computer information distribution network of an embodiment of the present invention shown in FIG. 23 after a reconfiguration event;

FIG. 25 is a schematic drawing of a topology of a computer information distribution network of an embodiment of the present invention, slightly different from the topology shown in FIG. 23, before a reconfiguration event;

FIG. 26 is a schematic drawing of a topology of a computer information distribution network of an embodiment of the present invention shown in FIG. 25 after a reconfiguration event;

FIG. 27 is a flow diagram of an embodiment of the present invention showing a Child Node's Propagation Routine;

FIG. 28 is a schematic drawing of another topology of a computer information distribution network of an embodiment of the present invention before a reconfiguration event;

FIG. 29 is a schematic drawing of a topology of a computer information distribution network of an embodiment of the present invention shown in FIG. 28 after a reconfiguration event;

FIG. 30 is a schematic drawing of another topology of a computer information distribution network of an embodiment of the present invention before a "complaint" regarding communications;

FIG. 31 is a flow diagram of an embodiment of the present invention showing a Grandparent's Complaint Response Routine.

FIGS. 32, 33A-33C, 34A-34C, 35A, 35B and 36 relate to "Registration Process" examples according to the present invention;

FIGS. 37 and 38 relate to a "Firewall Request / Firewall Status" example according to the present invention;

FIGS. 39A, 39B and 40 relate to a "Propagate Path / Down" example according to the present invention;

FIGS. 41A, 41B and 42 relate to a "Propagate NTM / Up" example according to the present invention;

FIGS. 43 and 44 relate to a "Banned List" example according to the present invention;

FIG. 45 relates to a "Node Interlacing" example according to the present invention; and

FIGS. 46A-46C relate to various distribution network topology examples according to the present invention.

Among those benefits and improvements that have been disclosed, other objects and advantages of this invention will become apparent from the following description taken in conjunction with the accompanying figures. The figures constitute a part of this specification and include illustrative embodiments of the present invention and illustrate various objects and features thereof (of note, the same reference numeral may be used to identify identical elements throughout the figures).

DETAILED DESCRIPTION OF THE INVENTION

Detailed embodiments of the present invention are disclosed herein; however, it is to be understood that the disclosed embodiments are merely illustrative of the invention that may be embodied in various forms. In addition, each of the examples given in connection with the various embodiments of the invention are intended to be illustrative, and not restrictive. Further, the figures are not necessarily to scale, some features may be exaggerated to show details of particular components. Therefore, specific structural and functional details disclosed herein are not to be interpreted as limiting, but merely as a representative basis for teaching one skilled in the art to variously employ the present invention.

Referring now to FIG. 3, an illustration of a linear propagation arrangement 10 of a computer network according to an embodiment of the present invention is shown. In this embodiment, the primary server node (or simply, server) 11 provides content data (e.g., streaming media) to user nodes 12 connected directly to it (sometimes referred to as "first level user nodes"). Each first level user node 12 has a second level user node 13 connected to it and each second level user node 13 has a third level user node 14 connected to it. The computer system at each first level user node 12 passes a copy of the content data received from server node 11 to the computer system at the second level user node 13 attached to such first level user node 12. The computer system at each second level user node 13 in turn passes the content data onto the computer system at the fourth level user node 14 attached to it.

As more fully discussed below, under this embodiment the computer systems at the server and user nodes may have distribution software installed in them which enables the nodes to be arranged as shown and for the computer systems to receive and re-transmit data.

The cascadingly connected arrangement of nodes (i.e., first level nodes are connected to the server, second level nodes are connected to first level nodes, third level nodes are connected to second level nodes and so on) shown in FIG. 3 takes advantage of the bandwidth available in certain nodes to essentially simultaneously receive and transmit data.

In a linear propagation arrangement, the effective distribution capacity of a server is in essence multiplied by the number of levels of nodes linked together. In the example of FIG. 3 (which example is intended to be illustrative and not restrictive), the distribution capacity of the server node is increased from 8 user nodes to 24 in just three levels.

Of note, many user nodes may have at least sufficient bandwidth (e.g., upstream bandwidth as well as downstream bandwidth) to receive data from one node and to re-transmit streams of data essentially simultaneously to two or more other nodes. This capacity could be used in another embodiment to set up a computer network with a cascadingly connected exponential propagation arrangement 16 (as shown in FIG. 4). As the name implies, an exponential propagation arrangement effectively increases the distribution capacity of a server exponentially. For example (which example is intended to be illustrative and not restrictive), with just three levels of user nodes, each having the capacity to retransmit two data streams, the distribution capacity of the server in FIG. 4 is increased from 8 user nodes to 56.

In another embodiment a distribution network may also be set up as a cascadingly connected hybrid linear/exponential arrangement 18, such as shown in FIG. 5.

Of note, the effective distribution capacity grows more quickly in a hybrid linear/exponential arrangement with each new level than does the distribution capacity in a linear propagation arrangement, and less quickly than does the distribution capacity in a pure exponential arrangement. However, any of these arrangements allows a server's distribution capacity to be greatly increased (e.g., with little or no additional investment in equipment for the server). Further, any other desired hybrid arrangement may be used.

Referring again to FIGS. 1-5, all of the connections between nodes are illustrated with arrows on each end. This is intended to signify that data flows in both directions between connected nodes. For example (which example is intended to be illustrative and not restrictive), a user node connected to a server may transmit data to the server indicating the identity of the user node, what the user node wants and/or other data, while the server node may transmits data confirming its identity and containing information and other content to the user node. In the remaining drawings such arrows may not be shown for the sake of simplicity. Further, it is noted that the bulk of data which is transmitted may be content (e.g., web pages, music files, streaming media), and such content will be understood to flow downtree from node to node in the drawings (i.e., in the direction from the root server to a first level node, to a second level node, to a third level node, etc.).

As can be understood from the above, an exponential propagation arrangement may create the most distribution capacity. However, while many (and perhaps even most) user nodes may have sufficient bandwidth to re-transmit acceptable quality multiple copies of the data, a number of user nodes may not have sufficient bandwidth to re-transmit more than one copy of the data with acceptable quality, and some user nodes may not have sufficient bandwidth to re-transmit even a single acceptable copy. Under these conditions, a distribution system employing the present invention may be configured, for example, as a hybrid propagation network. In such a system, a personal computer acting as a server node may reach many (e.g., hundreds, thousands or more) user nodes, even if the server node itself has capacity to transmit content data directly to only one other node.

In the following discussion it will be assumed that the system employing the present invention will take advantage of the capability of many user nodes to simultaneously re-transmit up to two copies of data (e.g., content data). However, it should be understood that the invention could take advantage of user nodes capable of simultaneous re-transmission of even higher numbers of copies.

In one embodiment the length of the distribution chains (i.e., the number of levels of user nodes linked through each other to the server) may be kept as small as possible to reduce the probability that the nodes at the ends of the chains will suffer a discontinuity of service. Thus, under this embodiment, in order to maximize the number of user nodes which may be connected to the server while trying to keep the length of the distribution chains as small as possible, the user nodes having the greatest bandwidth may be placed as far up uptree as possible (i.e., be placed as close as possible to the server, where "close" refers to network node level and not necessarily to geographical proximity).

In this regard, if an unreliable user node is placed near or at the end of a chain, few or no other user nodes would be affected by the unreliable user node's leaving the network. In contrast, if an unreliable user node were placed at or near the beginning of a chain (i.e., be far uptree), many other nodes would be affected by the unreliable node's departure. Because various embodiments of the invention may be used for appointment viewing of streaming media, in which an essentially continuous connection to an audiovisual content source may essentially be a necessity, it may be important (e.g., for this steaming media use) that the most reliable user nodes be placed high up in the chain (i.e., high uptree).

A user node may be deemed unreliable for any of a number of reasons. One example (which example is intended to be illustrative and not restrictive), is that the user node is connected to the Internet by lines having intermittent failures. Another example (which

example is intended to be illustrative and not restrictive), is that the user is merely sampling the content available on the network and intentionally disconnects the user node from the distribution network after discerning that he or she has no interest in the content.

Thus, under an embodiment of the present invention the user nodes may be positioned in the most advantageous positions, taking into account the dynamic nature of the distribution network, in which many user nodes may enter and leave the distribution network throughout the server's transmission of a streaming media show. In addition, the invention may help to preserve the viewing experience of users at user nodes positioned even at the end of a distribution chain (the server and/or user nodes may be enabled to perform the operations required to set up and maintain the distribution network by having data distribution software installed therein).

Referring now to a "User Node Arrival" example (which example is intended to be illustrative and not restrictive), it can be seen from FIGS. 3-5 that the distribution chains can be viewed as a plurality of family trees, each family tree being rooted to the server 11 through one of the first level nodes 12 (each node in the distribution network may have distribution software loaded in it which enables the node to perform the functions described below; before any new node may join the distribution network, such new node may also have such software loaded in it).

Of note, in the bulk of the examples discussed below, the network topologies have a branch factor of two (i.e., no user node is assigned more than two child nodes to be connected directly to it). A network topology with a branch factor of two may be referred to as a binary tree topology. It should be understood, of course, that the teachings set forth herein may be extended to network topologies having other branch factors, for example (which example is intended to be illustrative and not restrictive), branch factors of three, four or more.

In any case, FIG. 6 is a schematic drawing of a distribution network topology useful for describing the current "User Node Arrival" example. This FIG. 6 shows a distribution chain or family tree rooted to the server 11 through user node A, a first level user node 12 (the dashed lines represent connections from the server to other first level nodes or from node A to another user node). User node A could be thought of as a child node of the server and as a parent node for other user nodes connected directly to it. User node B, a second level user node 13, could be thought of as A's child. User nodes C and D, third level user nodes 14, may be thought of as B's children and A's grandchildren (and each other's siblings). User nodes E and F, fourth level user nodes 15, may be thought of as C's children (and each other's

siblings). User node G, also a fourth level user node 15, may be thought of as D's child. And user nodes E, F and G may be thought of as B's grandchildren and A's great grandchildren.

In the present example, whenever a new user node (or connection requesting user node) 19, such as node X in FIG. 6, seeks connection to the distribution network, it will first make a temporary connection to the server node (or a connection server, not shown) in order to begin the process for connecting to the distribution system. The server (or connection server) will discern from the user node a bandwidth rating (discussed below) appropriate to that node and, depending upon the available capacity of the server and any existing distribution chains, the server will either assign the new user node to a spot directly connected to the server or will provide the new user node with a connection path through a tree to the server.

FIG. 10 is a flow diagram associated with this example showing a Server's Connection Routine which is performed when a prospective child node seeks to join the distribution network. In step 101 the server performs the Server's Connection Instruction Routine (discussed below), in which the server determines what connection instructions to give to the new user node (or connection requesting user node). The server then goes to step 102 where it determines whether, as a result of the Server's Connection Instruction Routine, the prospective child node is being instructed to dock with the server. If the prospective child node is being instructed to dock with the server, then the server goes to step 103 in which the server would allow the new user node to dock with it, and the server would begin transmitting data (e.g., streaming media) directly to the new user node.

Of note, as referred to above, two different servers could be used - one to perform the server's connection routine and the other to transmit data (e.g., streaming media) - since both servers would be performing server functions, they will hereinafter sometimes be considered a single server for purposes of the description herein.

Of further note, while the prospective child node may be a new user node or a connection requesting user node, to simplify the following discussion such prospective child node may simply be referred to as a "a new user node".

In any case, if the new user node is not being instructed to dock directly with the server, then the server goes to step 104 in which it provides the new user node with an address connection list and disconnects the new user node from the server.

Essentially contemporaneously with the performance by the server of the Server's Connection Routine, the new user node is performing the Prospective Child Node's Request to Server for a Connection Routine. FIG. 11 is a flow diagram associated with this example

illustrating the Prospective Child Node's Request to Server for a Connection Routine. Upon making the temporary connection to the server, the new user node goes to step 111 in which it provides bandwidth rating and performance rating information to the server. It then proceeds to step 112 in which it receives connection instructions from the server. Then the new user node proceeds to step 113 to determine whether it has been instructed to dock directly with the server. If the answer is "yes," then the new user node proceeds to step 114 in which it erases any information which may have been in its ancestor database and, if the distribution software has a Return to Server Subroutine in it, resets the connection attempt counter to zero. The new user node then proceeds to step 115 in which it docks with the server and begins receiving data (e.g., streaming media). If the new user node is not being instructed to dock directly with the server, then the new user node goes to step 116 in which it receives the new connection address list from the server and loads such list into the user node's ancestor database and begins the Prospective Child Node's Connection Routine (discussed below). If the distribution software has a Return to Server Subroutine in it, the connection attempt counter is reset to one in step 116.

In the examples discussed in connection with FIGS. 6-9 and 12A-12C (which examples are intended to be illustrative and not restrictive), the server has determined that the new user node X will not be allowed to connect directly to the server. Also, for the purposes of these examples, all of the user nodes are presumed to be equally capable of simultaneously re-transmitting two copies of the content data and that the tree rooted through node A is the most appropriate tree through which node X should be connected to the server. In one embodiment the server may rely on chain length in determining to which particular user node, already in the distribution network, that node X should connect.

Referring again to the example of FIG. 6, the topology of the distribution network based on the most recent information available to the server at the moment that node X seeks to join the distribution network is shown. Path D-B-A-S (where "S" is the server) represents the shortest available path from an end of a chain back to the server (e.g., from a node level point of view and not necessarily from a geographical proximity point of view), and the server 11 gives that path information, or connection address list, to node X during step 104 of FIG. 10 (and node X receives such list during step 116 of FIG. 11). That is, node X will be given a connection address list with the URLs and/or IP addresses of each of nodes D, B, A and S. The distribution software in node X causes the path information to be stored in the ancestor portion (or ancestor database) 132 of node X's topology database 131 shown in FIG. 13. The ancestor database may include an ancestor list, a list of node X's ancestors' addresses

from node X's parent back to the server node. (FIG. 13 shows an example block diagram (which example is intended to be illustrative and not restrictive) showing the memory blocks into which the software used in connection with the present invention may partition a user node's memory 130.) Node X then attempts to contact node D first, the user node most distant from server 11 in the path. Note that when the server provides the D-B-A-S connection address to node X, the server is giving what it "believes" to be the complete path information going all the way back to the server. That is, subsequent to the most recent generation of the topological data, node D may have departed from the network, as may have one or more of its ancestors, resulting in a reconfiguration (discussed below) of at least a portion of the tree of which D was a part.

Referring now to FIG. 14 is a flow diagram associated with this example showing the Prospective Child Node's Connection Routine, the routine which a new user node, here node X, will go through in attempting to connect to a distribution chain (or tree) after receiving a connection address list (which node X has stored in the ancestor portion of its topology database) from the server or from a prospective parent node or during a reconfiguration event. In step 141, node X attempts to contact the first node on the connection address list. The first node, and only node, on the connection address list could be the server itself. Here node D is the first node on the list. Node X then proceeds to step 142 and determines whether the first node on the connection address list is still on-line and still part of the distribution network (in one example, if no response is received within a predetermined period of time, from the first node on the connection address list, the answer to the query in step 142 will be deemed to be no). If node D is on-line and still part of the distribution network, node X proceeds to step 143 in which node X inquires whether node D has room for node X. This inquiry may need to be made because the distribution network may have gone through a reconfiguration event resulting in node D's not having sufficient capacity to provide a copy of the content data to node X. If node D has room for node X, then node X proceeds to step 144 in which it connects (or docks) with node D and begins receiving content data from it. This is depicted in FIG. 7. Note that node X is now one of several level four nodes 15.

On the other hand, in step 142, if node D is not on-line (e.g., no response is received from node D within a predetermined period of time) or if node D is on-line but is no longer part of the distribution system (e.g., subsequent to the server's obtaining its topology data the user of the system at node D either caused his or her computer system to go off-line or to leave the distribution system, or there was a line failure causing the computer system to go off line), as depicted in FIG. 8, then node X goes to step 145 in which it deletes the first

address from the connection address list in node X's ancestor database (and, in one example, and for a purpose which will become clear when discussing reconfiguration events below, sets its reconfiguration flag buffer 136 to the lowest propagation rating). At this time node B, in the present example, becomes the first node on the connection address list. Then node X goes back to step 141 and repeats the routine described above. Note that because of node D's leaving the distribution network, a reconfiguration event was triggered which resulted in node G changing from a fourth level node 15 to a third level node 14.

In step 143, if the prospective parent node has no room for the new node (e.g., the capacity of the prospective parent node is fully occupied), the new node goes to step 146, in which it receives a new connection address list from the prospective parent. The prospective parent may then perform a Fully Occupied Parent's Connection Instruction Routine, discussed below in connection with FIG. 18, wherein it creates the new connection address list based on topology data obtained from its progeny. That new list may include the path back to the server through the prospective parent just in case, as discussed above, there are user node departures along the path.

Of note, by having a prospective parent node prepare the new connection address list as described in this example, the burden on the server is reduced and is distributed among the user nodes.

Still referring to the example depicted in FIG. 8, when node X gets to step 143, node B will respond that it has no room and node X will proceed to step 146. When node X performs step 146, the new connection address list it receives from node B will be G-B-A-S. Then node X proceeds to step 141 and repeats the routine from that point on. When the routine is performed on the topology shown in FIG. 8, step 143 will result in node G responding that it has room for node X. Node X will then perform step 144 and be connected to the distribution network through node G as shown in FIG. 9. Here, node X becomes a fourth level node 15.

The distribution software may include a Return to Server Subroutine, comprised of steps 151, 152 and 153 as shown in FIG. 15, as part of the Prospective Child Node's Connection Routine. This subroutine reduces the risk that a prospective child node would enter into an endless cycle of fruitless attempts to dock with nodes in a particular tree. If the answer to the query in step 143 is "no," then node X goes to step 151 in which it increments by one the connection attempt counter 135 in node X's memory. Then node X goes to step 152 in which it determines whether the connection attempt limit has been reached. In one example (which example is intended to be illustrative and not restrictive), the limit may be

preset at any number greater than one and may depend upon what the designer of a particular distribution network determines would be a reasonable amount of time for a node to attempt to make a connection on a particular tree, or a branch of a tree, before that node should be given an opportunity to obtain a completely new connection address list directly from the server. If the connection attempt limit has not been reached, then node X proceeds with step 146 as discussed above. If the connection attempt limit has been reached, then node X goes to step 153, in which it goes back to the server and begins the connection routine again as discussed above in connection with FIG. 6. If docking with a parent node is successful, then after step 144 node X performs step 154 in which the connection attempt counter is set to zero.

Referring now to FIG. 16, a flow diagram associated with this example illustrating the Prospective Parent Node's Connection Routine is shown. Using the example illustrated in FIGS. 6, 8 and 9, when node X queries node B in step 142 during the Prospective Child Node's Connection Routine, node B may begin performing the Prospective Parent Node's Connection Routine. In step 161, in response to node X's query, node B determines whether it is part of the distribution system node X seeks to join. If node B were not part of the distribution network, it would respond in the negative to node X's query and node B would be finished with the Prospective Parent Node's Connection Routine. In the example of FIG. 8, the answer is "yes" and node B proceeds to step 162.

In step 162 node B determines whether it has room for a new node. If the answer were "yes," node B would proceed to step 163 where it would allow node X to dock with it, and node B would begin transmitting to node X data (e.g., streaming media) originating from the server. In the example illustrated in FIG. 8, the answer is "no," and node B, acting as an instructing node, goes to step 164 where it performs the Fully Occupied Parent's Connection Instruction Routine (discussed below) and provides the prospective new child node (here node X) with a new connection address list. As noted above, the new connection address list may include the path back to the server through the node B (the prospective parent in this example) in the event that there are user node departures along the path, which departures may include node B.

After performing step 164 the temporary connection between node B and node X is terminated, and node X is sent on its way. In the example of FIGS. 8 and 9, the new connection address list is G-B-A-S. When node X approaches node G, node G performs the Prospective Parent Node's Connection Routine discussed above. In the example illustrated in FIGS. 8 and 9, node G allows node X to dock with it.

Referring now to a "Distribution Network Construction" example (which example is intended to be illustrative and not restrictive), it is noted that under this example the length of the distribution chains (i.e., the number of levels of user nodes linked through each other to the root server) is aimed to be kept as small as possible.

Assuming that all new user nodes have the same re-transmission capability (or disregarding the different re-transmission capabilities which different new user nodes may have), the user nodes would be distributed in this example through the first level until all the direct connection slots to the server were filled. Then as new user nodes sought connection to the distribution network, they would be assigned to connections with the first level nodes 12 until all the slots available on the first level nodes were filled. This procedure would be repeated with respect to each level as more and more new user nodes attempted to connect to the distribution network. In other words, the server, acting as an instructing node, would perform a Server's Connection Instruction Routine in which one step is determining whether there is room on the server for a new user node and, if so, the server would instruct the new user node to connect directly to the server. If there were no room on the server, then the server would perform the step of consulting its network topology database and devising a connection address list having the fewest number of connection links to a prospective parent user node. After performing the Server's Connection Instruction Routine, the server would either allow the new user node to dock directly with the server or send the new user node on its way to the first prospective parent user node on the connection address list.

In this example (which example is intended to be illustrative and not restrictive), a partially occupied potential parent node in a particular level (i.e., a prospective parent node already having a child but still having an available slot for an additional child node) may be preferred over unoccupied (i.e., childless) potential parent nodes on the same level. This may help to keep the number of interior nodes (i.e., the number of nodes re-transmitting data) to a minimum.

Alternatively, in another example (which example is intended to be illustrative and not restrictive), nodes that have zero children may be preferred over nodes that already have one child. While it is true that this may increase the number of repeat nodes, what the inventors have found is that by filling the frontier in an "interlaced" fashion, connecting nodes build up their buffers more quickly (allowing a reduction in the incidence of stream interruptions).

Referring now to FIGS. 12A-12C, these FIGS. illustrate, in this example, what happens when a partially occupied parent node is preferred over a parent childless node as a

destination address for a new user node (assuming all other factors are equal). FIG. 12A is a schematic diagram associated with this example showing a topology wherein nodes C and D, both third level nodes 14, have remaining capacity for one or more child nodes. Server 11 has the choice of sending new user node X to either node C, as shown in FIG. 12B, or node D, as shown in FIG. 12C, without increasing the length of the longest chain in the distribution network. However, user nodes are free to leave the distribution network at any time. With the topology shown in FIG. 12C, if either of nodes C or D leaves the network, there would be an effect on a child node. With the topology shown in FIG. 12B, there is a significant chance that if one of nodes C and D leaves the network, there would be no impact on any child nodes. That is, if it is node D that leaves the network, there is no child which would be made an orphan.

Of note, in discussing the examples illustrated in FIGS. 6-9 and 12A-12C, it has been presumed that all of the user nodes are equally capable of simultaneously re-transmitting two copies of the content data. However, the user nodes actually joining a distribution network will have various capabilities for re-transmitting data. Some will have no reliable capability because of hardware limitations. Other user nodes will have high nominal capability because of their hardware and type of Internet connection, but may have low reliability because of poor line conditions or the vagaries of the desires of the humans actually using the user nodes. For example, a user node having a T1 or faster connection to the Internet has a significantly large bandwidth, but is not reliable if the user (or viewer) is merely sampling the content available on the distribution network.

As noted above, in one example the user nodes having the greatest reliable capability should be placed as high up in a distribution chain as possible (i.e., as far uptree as possible) because they would have the ability to support the greatest number of child nodes, grandchild nodes and so on.

Thus, to differentiate the reliable capabilities of user nodes, a number of factors may be considered. Examples of such factors (which examples are intended to be illustrative and not restrictive), include following:

- One is time, that is, the number of seconds (or other units of time) since the node made its most recent connection to the network. Everything else being equal, a user node which has been continuously connected to the distribution network for a long period of time may be considered under this example to be likely more reliable

(either because of line conditions or user interest) than a user node which has been continuously connected to the network for a short period of time.

- Another factor is bandwidth rating, which may be determined, for example, by actual testing of the user node when it first attempts to connect to the server or a parent node (e.g., at initial connection time) or determined by the nominal bandwidth as determined by the type of connection made by the user node to the Internet. In one example (which example is intended to be illustrative and not restrictive), we will describe ratings based on nominal bandwidth as follows:

    o A user node with a 56 Kbits/sec dialup modem connection to the Internet is essentially useless for re-transmission of content data because of its small bandwidth. In this example, such a node is assigned a bandwidth rating of zero (0.0).

    o A user node with a cable modem or DSL (e.g., ADSL) connection to the Internet may be given, in this example, a bandwidth rating of one (1.0), because it is a common type of node and has a nominal upstream transmission bandwidth of 128 Kbits/sec, which is large enough to potentially re-transmit two acceptable quality copies of the content data it receives (assuming a 50Kbit/sec content data stream). Such capability fits well into a binary tree topology.

    o A full rate ISDN modem connection nominally has an upstream bandwidth of 56 Kbits/sec and a downstream bandwidth of 56 Kbits/sec, which would potentially support acceptable quality re-transmission of a single copy of the content data stream (assuming a 50Kbit/sec content data stream). For this reason, a user node with a full rate ISDN modem connection to the Internet may be given, in this example, a bandwidth rating of one-half (0.5), or half the rating of a user node connected to the Internet by a DSL (e.g., ADSL) or cable modem connection.

    o User nodes with T1 or greater links to the Internet should be able to support more (e.g. at least twice as many) streams as DSL (e.g., ADSL) or cable modems, and therefore may be given, in this example, a bandwidth rating of two (2.0). In the event that in a distribution network parent nodes

may be assigned more than two child nodes directly connected thereto, bandwidth ratings greater than 2.0 may be assigned under this example to Internet connections having greater bandwidth than T1 connections.

- In another example (which example is intended to be illustrative and not restrictive), other bandwidth networks (e.g., 50 Kbit/sec and 100 Kbit/sec) may be supported.
- In another example, the bandwidth needed for a node to be "repeat capable" may depend on the particular network to which it is attached (upon initial connection the connecting node may first test its upstream bandwidth and then report that bandwidth to the network server; the server may determine whether the node is "repeat capable" or not.
- In another example (which example is intended to be illustrative and not restrictive), a node may be deemed "repeat capable" if the node is not firewalled (e.g., port 1138 is open) and the node has sufficient bandwidth to retransmit two copies of the incoming stream.
- A third factor is performance. A user node's performance rating may, in one example, be zero (0.0) if it is disconnected as a result of a Grandparent's Complaint Response Routine (discussed below in connection with FIG. 31). Otherwise, the user node's performance rating, in this example, may one (1.0).

Further, in this example (which example is intended to be illustrative and not restrictive), a user node's utility rating may determined by multiplying connection time by performance rating by bandwidth rating. That is, in this example:

- Utility Rating = Time x Performance Rating x Bandwidth Rating.

Referring once again to FIG. 13, it is seen that in this example information regarding a user node's time connected to the network, bandwidth rating, performance rating, utility rating and potential re-transmission rating (discussed below) may be stored in the user node's elapsed time buffer 137, bandwidth rating buffer 138, performance rating buffer 139, utility rating buffer 121 and potential re-transmission rating buffer 122, respectively.

In another example (which example is intended to be illustrative and not restrictive), nodes entering the network may be judged to be either "repeat capable" or "non-repeat capable" (wherein non-repeat capable nodes may be called "bumpable" nodes). Repeat

capability may be based on: (1) Upstream bandwidth (e.g., tested at initial connection); and (2) the firewall status, opened or closed, of a node. In one specific example, if a node is either firewalled or has upstream bandwidth less than (approx.) 2.5 times the video streaming rate, that node will be deemed bumpable. All nodes joining the network, whether bumpable or non-bumpable, may be placed as close to the server as possible. However, for the purpose of placement (as described for example in the Universal Connection Address Routine of Figure 20), a repeat capable node can bump a bumpable node when being placed in the network.

Referring now to FIG. 17 a flow diagram associated with this example illustrating the Server's Connection Instruction Routine is shown. This routine need not necessarily rely on the new user node's utility rating. This routine may rely on the potential re-transmission rating of a user node, which is arrived at by multiplying the user node's performance rating by its bandwidth rating. That is, in this example:

- Potential Re-transmission Rating = Performance Rating x Bandwidth Rating.

Thus, in this example, the server would want to put those user nodes with the highest potential re-transmission rating as close to the server in a distribution chain as possible (i.e., as high uptree as possible) because they have the greatest likelihood of being able to re-transmit one or more copies of content data. On the other hand, a potential re-transmission rating of zero in this example indicates that a user node has no ability to (or little expected reliability in) re-transmitting even one copy of content data (the server in this example would want to put a user node with a zero rating as far as reasonably possible from the server in a distribution chain (i.e., as low downtree as possible)). For the purpose of ease of discussion, in the flow diagram example of FIG. 17 illustrating the Server's Connection Instruction Routine, the server is concerned about whether the potential re-transmission rating is zero (i.e., either one or both of the performance rating and bandwidth rating is zero) or greater than zero (i.e., both the performance rating and the bandwidth rating are greater than zero).

In step 171 the server node interrogates the new user node (or connection requesting node) for its bandwidth rating and performance rating. If the new user node is really new to the distribution network, or if it has returned to the server because all of the user node's ancestor nodes have disappeared from the network, the new user node's performance memory will contain, in this example, a performance rating of 1.0 (e.g., the default rating). However, if the new user node has been dispatched to the server for a new connection to the network

because the new user node had failed to provide content data to one of its child nodes, then, in one example, its performance memory will contain a performance rating of zero.

In step 172, the server determines, in this example, whether the potential re-transmission rate of the connection requesting node is greater than zero (i.e., whether both the bandwidth rating and the performance rating are greater than zero, or, if only the bandwidth rating is considered, whether the bandwidth rating is greater than zero (i.e., the connection requesting node is a high-bandwidth node)). If the answer is "yes," then the server goes to step 173 in which the server determines whether it has room for the new user node. If the answer to the query in step 173 is "yes," then the server goes to step 174 in which it instructs the new user node to connect directly to the server. Then the server goes to step 102 in the Server's Connection Routine (see FIG. 10).

If the answer to the query in step 173 is "no" (i.e., the server does not have the capacity to serve the new user node directly), then the server goes to step 175. In step 175, the server, acting as an instructing node, consults its network topology database and devises a connection address list having the fewest number of connection links to a prospective parent node. That is, the server checks its information regarding the nodes in the level closest to the server (i.e., the highest uptree) to determine whether there are any potential parent nodes with space available for the new user node. If there are no potential parent nodes with space available, then the database is checked regarding nodes in the level one link further from the server, and so on until a level is found having at least one potential parent node with space available for the new user node. That is, the server determines which parent node with unused capacity for a child node is closest to the server (i.e., in the highest level, with the first level being the highest), and devises the connection address list from such prospective parent node to the server. The server then goes to step 102 (shown in FIG. 10).

In another embodiment of the invention, the server could skip step 172 and go directly to step 173 (or it could go to step 172 and, if the answer to the query in step 172 is "no" (i.e., either one or both of the bandwidth rating and the performance rating are zero, or, if only the bandwidth rating is considered, whether the bandwidth rating is zero (i.e., the connection requesting node is a low-bandwidth node)), the server could go to step 175). However, this could prevent the server from loading up the highest levels of the distribution chains with nodes capable of re-transmitting at least one acceptable copy of the content data (in contrast to if the server does perform step 172 and does goes to step 176 if the answer to the query in step 172 is "no.").

In step 176 the server consults its network topology database and devises a connection address list having the greatest number of connection links to a prospective parent node. That is, the server checks its information regarding the nodes in the level furthest from the server (i.e., the lowest downtree) to determine whether there are any potential parent nodes with space available for the new user node. If there are no potential parent nodes with space available, then the database is checked regarding nodes in the level one link closer to the server, and so on until a level is found having at least one potential parent node with space available for the new user node. In this manner, user nodes having limited or no reliable re-transmission capability may be started off as far from the server as possible and will have a reduced effect on the overall capacity of the distribution network.

As indicated above, and as discussed more fully below, one or more reconfiguration events may have transpired since the server's topology database was last updated. As a result, the first prospective parent node which is actually present on the distribution network for the new user node to contact may not have room for the new user node. By way of example (which example is intended to be illustrative and not restrictive), when node X tries to join the distribution network having the topology shown in FIG. 12A, the server provides node X with the following connection address list: C-B-A-S. If node C had disappeared from the network between the last update of the server's topology database and node X's attempting to contact node C, then node E, by virtue of a reconfiguration event, would be connected, in this example, to node B as shown in FIG. 12D. Then node X, in performing the Prospective Child Node's Connection Routine discussed in connection with FIGS. 14 and 15, would contact node B. Node B, in the Prospective Parent Node's Connection Routine, discussed in connection with FIG. 16, would have to answer the query of step 162 in the negative and go to step 164, in which it performs the Fully Occupied Parent's Connection Instruction Routine.

Referring now to FIG. 18, a flow diagram associated with this example illustrating that routine is shown. It is similar to the Server's Connection Instruction Routine. Since the fully occupied parent node has already determined that it has no room for the new user node (or connection requesting user node), the Fully Occupied Parent's Connection Instruction Routine does not need to include a step in which a determination is made regarding whether there is room for the new user node (or connection requesting node). In the Fully Occupied Parent's Connection Instruction Routine the fully occupied parent node, acting as an instructing node, first performs step 181 in which it interrogates the new user node for its bandwidth rating and performance rating. In step 182, the fully occupied parent node determines whether the potential re-transmission rate of the new user node is greater than

zero. (If only the bandwidth rating is considered, then it determines whether the bandwidth rating is greater than zero (i.e., the connection requesting node is a high-bandwidth node).) If the answer is "yes," then the fully occupied parent node goes to step 183 in which the fully occupied parent node consults its topology database, which contains the latest information available to that node regarding the links from the fully occupied parent node back to the server and regarding the fully occupied parent node's own progeny (i.e., its children, grandchildren etc.,) and devises a new connection address list having the fewest number of connection links to a new prospective parent node. That is, the fully occupied parent node checks its information regarding the nodes in the level closest to the fully occupied parent node (but not closer to the server) to determine whether there are any potential parent nodes with space available for the new user node. If there are no potential parent nodes with space available, then the database is checked regarding nodes in the level one link further from the fully occupied parent node (and further from the server), and so on until a level is found having at least one potential parent node with space available for the new user node. That is, the fully occupied parent node determines which new prospective parent node with unused capacity for a child node is closest to the fully occupied parent node, and devises the connection address list from such new prospective parent node through the fully occupied parent node and on to the server. The fully occupied parent node then goes to step 184 in which it provides the new user node with the new connection address list and disconnects the new user node from the fully occupied parent.

If the answer to the query in step 182 is no (i.e., either (i) one of the bandwidth rating and performance rating is zero or (ii) if only the bandwidth rating is considered, the bandwidth rating is zero (i.e., the connection requesting node is a low-bandwidth node)), the fully occupied parent node goes to step 185. In step 185 the fully occupied parent node consults its network topology database and devises a connection address list having the greatest number of connection links to a new prospective parent node. That is, the fully occupied parent node checks its information regarding the nodes in the level furthest from it (and farther from the server than it is) to determine whether there are any potential parent nodes with space available for the new user node. If there are no potential parent nodes with space available, then the database is checked regarding nodes in the level one link closer to the fully occupied parent node, and so on until a level is found having at least one potential parent node with space available for the new user node. As discussed above, this helps assure that new user nodes having limited or no reliable re-transmission capability are started off as far from the server as possible. After the fully occupied parent node devises the connection

address list from such new prospective parent node through the fully occupied parent node and on to the server, the fully occupied parent node then goes to step 184 where it performs as discussed above.

In another embodiment of the invention, the distribution software could be designed such that a fully occupied parent performs an abbreviated Fully Occupied Parent's Connection Instruction Routine, in which steps 181, 182 and 185 are not performed. That is, it could be presumed that the server has done the major portion of the work needed to determine where the new user node should be placed and that the fully occupied parent user node need only redirect the new user node to the closest available new prospective parent. In such event, only steps 183 and 184 would be performed.

In the example discussed above in which node C had disappeared from the network when new user node X had been given, by the server, connection address C-B-A-S, and in which node B is a fully occupied parent node as shown in FIG. 12D, node B would appear to have the choice of devising either connection address list D-B-A-S or E-B-A-S regardless of whether the full or abbreviated Fully Occupied Parent's Connection Instruction Routine were performed.

In this regard, to determine which connection address list to choose when there are two or more prospective parent nodes in a particular level having space for a new user node, the distribution software could have an additional subroutine as part of steps 175, 176, 183 and 185. An example of this subroutine, called the Multi-Node Selection Subroutine is illustrated in FIG. 19.

In step 191 the server or fully occupied parent node deciding where to send a new user node determines whether any of the potential new parent nodes is partially occupied. As discussed earlier, in one example a partially occupied potential parent node may be preferred over an unoccupied potential parent node. In this case, if any of the potential parent nodes is partially occupied, then the server or fully occupied parent node goes to step 192. In step 192 the partially occupied prospective parent node with the highest utility rating is selected as the new prospective parent node. If there were only a single partially occupied potential parent node, then that node is selected.

If in step 191 it is determined that there are no partially occupied potential parent nodes, then the server or fully occupied parent node goes to step 193. In step 193 the unoccupied prospective parent node with the highest utility rating is selected as the new prospective parent node.

In another example (which example is intended to be illustrative and not restrictive),the software engineer could have step 193 follow an affirmative response to the query in step 191 and step 192 follow a negative response; in such event, unoccupied prospective parent nodes would be selected ahead of partially occupied prospective parent nodes (all other things being equal, it may be advantageous to select nodes that have zero children over nodes that already have one child; while it is true that this may increase the number of repeat nodes, what the inventors have found is that by filling the frontier in an "interlaced" fashion, connecting nodes build up their buffers more quickly (allowing a reduction in the incidence of stream interruptions).

After whichever of steps 192 and 193 is completed, the server or fully occupied parent node returns to the step from which it entered the Multi-Node Selection Subroutine (i.e., step 175, 176, 183 or 185), and completes that step.

So, in the example shown in FIG. 12D, where node C left the network thereby leaving node B with the chore of devising a new connection address list for node X, node B would perform the Fully Occupied Parent's Connection Instruction Routine. Regardless of the bandwidth and performance ratings of node X, node B would be choosing between nodes D and E in the third level. In step 191 node B would determine that neither D nor E is partially occupied, and therefore node B would go to step 193. Assuming that nodes E and D have equal bandwidth and performance ratings and that node D was connected to the network longer than node E, node D would be selected because it would have the higher utility rating since it was connected to the network longer than node E. Node B would then go to step 194 and then return to the step from which it entered the Multi-Node Selection Subroutine. When node B returns to step 183 or 185, it completes that step and moves on to step 184. In that step, node B provides new user node X with new connection address list D-B-A-S and node X connects to the distribution network as shown in FIG. 12E.

Referring now to an "Alternative Distribution Network Construction" example (which example is intended to be illustrative and not restrictive), it is noted that under this example the length of the distribution chains (i.e., the number of levels of user nodes linked through each other to the server) is aimed to be kept as small as possible.

Also under this example, user nodes having the highest bandwidth capabilities are aimed to be kept closer to the server (e.g., in order to allow the greatest expansion of the distribution system). However, it is possible that zero bandwidth rated nodes may nevertheless appear relatively far uptree (thereby stunting the growth of that chain). The following method may be used in constructing the distribution network both by servers and

by prospective parents which are actually completely occupied, either of which may be thought of as an instructing node (that is, software enabling the routines discussed below could be installed on servers and user nodes alike).

More particularly, in the distribution system of this example, each child node reports directly to (or is tested by) its parent node with respect to information relating to the child node's bandwidth rating, performance rating, potential re-transmission rating and utility rating. In turn, each parent node reports all the information it has obtained regarding its child nodes on to its own parent node (a parent node also reports to each of its child nodes the address list from that parent back to the server, which list forms what may be referred to as the ancestor portion of the topology database – in addition, a parent node reports to each of its child nodes the addresses of their siblings). The reports occur during what may be referred to as a utility rating event. Utility rating events may occur, for example, on a predetermined schedule (e.g., utility rating events may occur as frequently as every few seconds). As a result of all the reporting, each node stores in its topology database the topology of the tree (including all its branches) extending below that node, and the server stores in its topology database the topology of all the trees extending below it. This may be referred to as the descendant portion (or descendant database) 133 of the topology database (see FIG. 13). The descendant database of a particular node may include a descendant list, a list of the addresses of all the nodes cascadingly connected below that particular node. Included in the topology database information may be the utility ratings of the nodes below the node in which that particular topology database resides.

After each reporting event in this example, each parent node (including the server), acting as an instructing node, devises two lists of prospective (or recommended) parent nodes. In this example, the first list, or Primary Recommended Parent List ("PRPL"), stored in the Primary Recommended Parent List buffer 123 (see FIG. 13), lists all the nodes in the descendant portion of that node's topology database which have bandwidth available to support another child node (in one example (which example is intended to be illustrative and not restrictive), in a binary tree system, all nodes in the descendant portion of the topology database having (i) a bandwidth rating of at least one and (ii) less than two child nodes would be listed). They would be listed with those node's which are closest to the node in which that particular topology database resides at the top of the list, and those nodes which are in the same level would be ranked such that the node with the highest utility rating would be listed first, the node with the second highest utility rating would be listed second and so on. By way of example (which example is intended to be illustrative and not restrictive), the PRPL of a

second level node would list a third level node with available bandwidth ahead of a fourth level node with available bandwidth even if the fourth level node's utility rating were higher than that of the third level node.

The second list in this example, or Secondary Recommended Parent List ("SRPL"), stored in the Secondary Recommended Parent List buffer 124 (see FIG. 13), lists all the nodes in the descendant portion of that node's topology database which have the ability to re-transmit content data to child nodes but are fully occupied, and at least one of its child nodes is incapable of re-transmitting content data to another child node (in one example (which example is intended to be illustrative and not restrictive), in a binary tree system, all nodes in the descendant portion of the topology database having (i) a bandwidth rating of at least one and (ii) at least one child node having a bandwidth rating less than one (i.e., being incapable of re-transmitting content data to two child nodes) would be listed). Like the nodes in the PRPL, the nodes in the SRPL would be listed with those node's which are closest to the node in which that particular topology database resides at the top of the list, and those nodes which are in the same level would be ranked such that the node with the highest utility rating would be listed first, the node with the second highest utility rating would be listed second and so on.

Of note, in this example the SRPL lists those parent nodes having the growth of their branches (i.e., their further progeny) blocked or partially blocked by a low-bandwidth child node. This may lead to an unbalanced growth of the distribution system, and a limitation on the total capacity of the system.

Of further note, to the extent that a node (including a server) has room for another child node or is the parent of a low bandwidth node, it may be listed on its own PRPL or SRPL.

Referring now to FIGS. 20A and 20B, an example flow diagram illustrating what may be referred to as the Universal Connection Address Routine is shown. As indicated above, a server or a fully occupied prospective parent node receiving a connection request may be referred to as an "instructing node." When a new node (here node N) approaches an instructing node, the instructing node performs step 201, it receives a connection request. It then goes to step 202 in which it interrogates or tests node N to determine whether it is a low-bandwidth node ("low-bandwidth" may mean, for example, either or both: (1) a node that is incapable of broadcasting two copies of its incoming stream on to its children; and (2) a node that is incapable of rebroadcasting its stream because it is firewalled). In the binary tree system of one example, a low-bandwidth node is a node with a bandwidth rating less than

one. If node N is a low bandwidth node, the instructing node proceeds to step 203 in which the instructing node determines whether there are any available prospective parent nodes which are not fully occupied. Sometimes the distribution network may be fully occupied. If it is, the instructing node's PRPL will be empty. If it is empty, the response to the query in step 203 would be yes. Then the instructing node goes to step 209 in which the new node is sent back to the server to start the connection process from the beginning. If the response to the query in step 203 is no, then the instructing node goes to step 204 in which it selects a prospective (or recommended) parent node for node N. The instructing node then moves on to step 205 in which it consults its topology database and devises a connection address list from the recommended parent node back to the server, and provides such connection address list to node N. (If the instructing node is a user node, then the connection address list leads back to the server through the instructing node.) At this point, node N performs the Prospective Child Node's Connection Routine discussed above in connection with FIG. 14.

As part of step 205, the instructing node inserts node N into its topology database as a child of the recommended parent node. It does this because other new nodes may seek connection instructions prior to the next utility rating event (i.e., before reports providing updated information regarding the topology of the distribution network are received by the instructing node), and such new nodes should not be sent to a prospective parent which the instructing node could know is fully occupied. In this regard, the instructing node then goes to step 206 in which it checks its topology database to determine whether the recommended parent, with node N presumptively docked to it as a child node, is fully occupied (in the example here of a binary tree network, whether it has two child nodes). If the answer is no, then the instructing node has finished this routine. If the answer is yes, then the instructing node goes to step 207 in which it removes the recommended parent from the PRPL and inserts it into the SRPL, and the instructing node has finished the routine.

If the answer to the query in step 202 is no (e.g., node N is not a low-bandwidth node; in the binary tree network example it is a high-bandwidth node capable of re-transmitting content data to two child nodes), the instructing node moves on to step 208. There it determines whether both the PRPL and SRPL are empty (which may occur under certain circumstances, such as, for example, when the number of levels in a distribution system is capped and at least all the nodes on all but the last level are fully occupied with high-bandwidth nodes). If so, the instructing node goes to step 209 in which the new node is sent back to the server to start the connection process from the beginning. If the response to the query in step 208 is no, then the instructing node goes to step 210 in which it selects a

prospective (or recommended) parent node for node N from the PRPL and an alternate recommended parent node from the SRPL (if either the PRPL or SRPL is empty, the instructing node will make a "nil" selection from that list – the instructing node knows from step 208 that at least one of the lists will not be empty). The instructing node then goes to step 211 in which it determines whether the alternate recommended parent node is closer to the server (i.e., higher uptree) than the recommended parent node derived from the PRPL. If the alternate recommended parent node is on the same level as, or on a lower level than the recommended parent node derived from the PRPL (or if the selection from the SRPL is nil), then the answer to the query in step 211 is no.

In such event, the instructing node goes to step 212 in which it consults its topology database and devises a connection address list from the recommended parent node back to the server, and provides such connection address list to node N (if the instructing node is a user node, then the connection address list leads back to the server through the instructing node).

As in step 205 discussed above, at this point, node N performs the Prospective Child Node's Connection Routine discussed above in connection with FIG. 14, and in step 212, the instructing node inserts node N into its topology database as a child of the recommended parent node.

The instructing node moves to step 213 in which it adds node N to an appropriate position on the PRPL. It does this because, as a result of step 202, it knows that the node N is capable of re-transmitting content data to its own child nodes.

The instructing node then goes to step 214 in which it checks its topology database to determine whether the recommended parent, with node N presumptively docked to it as a child node, is fully occupied (in the example here of a binary tree network, whether it has two child nodes). If the answer is no, then the instructing node has finished this routine. If the answer is yes, then the instructing node goes to step 215 in which it removes the recommended parent from the PRPL because it is now deemed to not be an available prospective parent node.

Then the instructing node goes to step 216 in which it consults its topology database to determine whether any of the recommended parent node's children is a low-bandwidth node (in this example, knowing that node N is not a low-bandwidth node and knowing that the recommended parent node has two child nodes, the question is whether the child node other than node N is a low-bandwidth node.) If the answer is no (i.e., all the recommended parent node's children are high-bandwidth nodes), then the instructing node has finished the routine.

If the answer is yes, then the growth of the recommended parent node's line of progeny is partially blocked by a low-bandwidth child node. The instructing node moves on to step 217 in which it adds the recommended parent to the SRPL.

If the answer to the query in step 211 is yes (i.e., the alternate recommended parent (selected from the SRPL) is: (i) on a higher level than the recommended parent (selected from the PRPL) or (ii) the selection from the PRPL is nil), then the instructing node moves to step 218. In that step the instructing node consults its topology database to determine: (i) which of the alternate recommended parent node's child nodes is a low-bandwidth node or (ii) if they both are low-bandwidth nodes, which of the child nodes has been connected to the system a shorter period of time. In one example (which example is intended to be illustrative and not restrictive), the instructing node may send a disconnect signal to that child node with instructions to return to the server to start the connection process from the beginning (as a new user node (or connection requesting user node)). In another example (which example is intended to be illustrative and not restrictive), the bumped node may climb its path to its grandparent which gives it a new connection path – rather than returning to the root server (doing so typically means that the incoming node that bumped the non-repeat-capable node ends up being that node's new parent).

The instructing node moves on to step 219 in which it consults its topology database and devises a connection address list from the alternate recommended parent node back to the server, and provides such connection address list to node N (if the instructing node is a user node, then the connection address list leads back to the server through the instructing node).

As in steps 205 and 212 discussed above, at this point, node N performs the Prospective Child Node's Connection Routine discussed above in connection with FIG. 14, and in step 219, the instructing node inserts node N into its topology database as a child of the alternate recommended parent node.

The instructing node moves to step 220 in which it adds node N to an appropriate position on the PRPL. It does this because, as a result of step 202, it knows that the node N is capable of re-transmitting content data to its own child nodes.

The instructing node then goes to step 221 in which it checks its topology database to determine whether all the child nodes of the alternate recommended parent are high-bandwidth nodes. If the answer is no, then the instructing node has finished this routine. If the answer is yes, then the instructing node goes to step 222 in which it removes the alternate recommended parent from the SRPL because it is now deemed to not be an available

alternative prospective parent node since the growth of its progeny line is not even partially blocked by one of its own children. At this point the instructing node has finished the routine.

With the routines discussed in connection with the above examples, the distribution network will be built with each new node assigned to the shortest tree (or chain), and those with the fewest number of links between it and the server. However, low-bandwidth nodes, which would tend to block the balanced growth of the distribution network, would be displaced by high-bandwidth nodes and moved to the edges of the network where they would have reduced effect on the growth of the network.

Referring now to FIG. 21, this FIG. may be used as an example to illustrate how the Universal Connection Address Routine works. It shows primary server 11 with node A as first level node 12, its child nodes B and C as second level nodes 13, their child nodes D, E, F and G as third level nodes 14, node D and G's child nodes H, I, J and K as fourth level nodes 15, and node I's child nodes L and M as fifth level nodes 17. Assume for this example that all available docking links directly to the server 11 are occupied by high-bandwidth nodes, and that all trees are at full capacity other than that rooted to the server 11 through first level node A. The utility rating for each node is set forth in FIG. 21 under its letter designation. Low-bandwidth nodes (in this example, those nodes having a bandwidth rating less than one) are shown with a bandwidth rating of "Ω," which indicates in this example that such nodes will not have any child nodes assigned to them. Nodes A, B, C, D, E, G, I, J and K are high-bandwidth nodes (or repeater nodes (i.e., they are capable of re-transmitting the content data they receive from their respective parent nodes to their respective child nodes, if any)).

A new user node (or other connection requesting user node), here referred to as "node N", may be directed to node A as a result of various reconfiguration events. When node A receives a connection request from node N, node A will, in this example, either consult both of its PRPL and SRPL buffers, if node N is a high-bandwidth node, or consult just its PRPL buffer (if node N is a low-bandwidth node).

Nodes F, H, L, and M will not appear on any list in this example since they are low-bandwidth nodes. Nodes B, C, D, G, I and A itself will not appear on the PRPL in this example since these nodes are fully occupied. However, nodes C, D, and I will appear on the SRPL in this example because they each have at least one low bandwidth child node.

Using the rules discussed above (i.e., ranking prospective parent nodes by level and within levels by utility rating) the PRPL would be as follows in this example:

- PRPL

- E (a third level node with available capacity)
  - J (the fourth level node with available capacity having the highest utility rating)
- K (a fourth level node with available capacity)

The SRPL in this example would be as follows:

- SRPL
  - C (the fully occupied second level node with a low-bandwidth child node)
  - D (the fully occupied third level node with a low-bandwidth child node)
  - I (the fully occupied fourth level node with a low-bandwidth child node)

If node N is a low-bandwidth node, node A will give, in this example, node N the following as its new connection address list:

- E-B-A-S.

Since node E would not have two child nodes, it would remain on the PRPL.

If node N is a high-bandwidth node, node A will compare (step 211 of FIG. 20B) the first node on the PRPL (the recommended parent node) with the first node on the SRPL (the alternate recommended parent node). Here, node C, the first node on the SRPL is a higher level node than node E, the first node on the PRPL. So, node A will send a disconnect signal (step 218 of FIG. 20B) to node F, node C's low-bandwidth child node. Then it will provide node N with the following new connection address list and add node N to the PRPL (step 219 of FIG. 20B):

- C-A-S.

Since node C would now have two high-bandwidth child nodes (nodes N and G), node C would be removed from the SRPL (step 222 of FIG. 20B).

Further, in this example wherein node N is a high-bandwidth node, when node F returns to the server 11 (or, in another example, goes to Node A) for a new connection, the

server (or Node A) will also use the Universal Connection Address Routine (Node F may go to Node A rather than the server because (in one example) when a node is "kicked" it may climb its internal connection path rather than jumping directly to the root server). Since node F is a low-bandwidth node, the server (or Node A) will give node F the following as its new connection address list:

- E-B-A-S.

FIG. 22 illustrates the new topology. As can be seen, absent intervening events, low-bandwidth node F will end up moving down from the third level to the fourth level and the bandwidth capacity of the third level will increase from six child nodes to eight (its maximum in this binary tree example). The potential bandwidth capacity of the fourth level would also increase, from ten child nodes to twelve.

Referring now to a "Distribution Network Reconfiguration" example (which example is intended to be illustrative and not restrictive), it is noted that under this example the user nodes are free to exit the distribution network at will. And, of course, user nodes may exit the distribution network due to equipment failures between the user node and the communication network (the Internet in the examples discussed herein). This was first discussed in connection with FIGS. 10A-E.

Under this example the present invention may handle the reconfiguration required by a user node departure by having certain user nodes in the same tree as the departing node perform a Propagation Routine. The results of the propagation routine of this example are illustrated in FIGS. 23 and 24. There a tree rooted to the server through first level user node P is illustrated. Node P has two child nodes, second level nodes Q and X. Through node X, P has two grandchild nodes, third level nodes A and B. Based on the relative utility ratings of nodes Q and X, P has sent signals to its children instructing them to set the color of the flag represented by the information in their respective reconfiguration flag buffers to "green" and "red," respectively. The use of colors as designators is merely discretionary. In reality they represent relative ratings of siblings for purposes of determining their roles in a reconfiguration event, and may be referred to as "propagation ratings." In addition, instead of colors, numbers (e.g., 1 and 2) could be used, as well as any of many other nomenclature schemes to describe the grades of propagation ratings. The "red" "green" ratings discussed herein shall be deemed representative of all such schemes. The number of grades of propagation ratings assigned by a parent node may be equal to the number of children each

parent node has. For example, for a branch factor "n" system, the maximum number of grades of propagation ratings may be "n." Since the distribution network in the examples discussed herein is a binary tree distribution network, a parent node will be required to assign at most up to two grades of propagation ratings.

In the example shown in FIG. 23, node P, during the most recent utility rating measurement event, discerned that node Q has a higher utility rating than node X, and therefore P has assigned node Q a green rating, represented by the solid-lined circle surrounding the letter Q in FIG. 23. P has assigned node X a red rating, represented by the dashed-lined circle surrounding the letter X in FIG. 23. In a similar manner, node X has assigned green and red ratings to third level nodes A and B, respectively. In this example (which example is intended to be illustrative and not restrictive), at the same time that a parent node assigns a propagation rating to a child node, it may also provide to such child node the address of the child node's sibling, if there is any. A child node may store information about its sibling (or siblings) in the sibling portion (or sibling database) 134 of its topology database (see FIG. 13). The sibling database includes a sibling list, a list of the addresses of a node's siblings (the data relating to the siblings' addresses may also contain information regarding the propagation ratings of the siblings (e.g., in the event that the distribution system has a branch factor greater than two)). In the example shown in FIG. 23, nodes Q and X know that they are each other's siblings and nodes A and B know that they are each other's siblings.

In another example (which example is intended to be illustrative and not restrictive), nodes do not store information about their siblings. In other words, in this example, a node generally does not know who its sibling is. Instead, cross connections between a red node and its green sibling are implemented via a "priority join" message sent to a red child that includes the IP address of the green sibling as the first entry in the connection path list.

In the event that node X were to leave the distribution network, nodes A and B would of course stop receiving content data from node X. Nodes A and B would consult their topology databases for the address of their grandparent, and each would interrogate node P for instructions. Since node X is out of the distribution network, node P would send out a reconfiguration event signal (sometimes referred to as a "propagation signal") to the nodes which had been node X's children (i.e., the nodes which were node P's grandchildren through node X) and node P would send an upgrade propagation rating signal to its remaining child, here node Q.

In response to the upgrade propagation rating signal, node P's remaining child, node Q, would set its reconfiguration flag buffer 136 (see FIG. 13) to the next highest available propagation rating grade. Since in the example illustrated in FIG. 23, node Q's reconfiguration flag buffer already is set for the highest propagation rating grade (here green), node Q could either do nothing or reset its propagation rating in response to the upgrade propagation rating signal. The result would be the same, node Q's propagation rating would remain green. If node Q's propagation rating were red, then it would set its propagation rating to green in response to the upgrade propagation rating signal. Node Q would do nothing else in response to the upgrade propagation rating signal (note that as a matter of design choice, the software engineer could have the parent of the departed node (here node P is the parent of departed node X) send no upgrade propagation rating signal to its remaining child node (here node Q) if its remaining child node already is at the highest propagation rating).

In this example, the recipients of the propagation signal (i.e., the children of the missing node) would respond thereto as follows. First they would check their own respective propagation ratings. If the propagation signal's recipient has the highest propagation rating grade (here green), it would reset its propagation rating to the lowest rating grade (here red); re-transmit the propagation signal to its own child nodes (if there are any); disconnect the child node which did not have the highest propagation rating prior to its receipt of the propagation signal if the propagation signal's recipient with the green rating has more than one child node; allow its sibling to dock with it as a child node for the purpose of transmitting content data to that child node; and dock (or remained docked), for purposes of receiving content data, with the node sending the propagation signal to it (in systems having a branch factor of more than two, the propagation signal recipient whose propagation rating had been the highest would disconnect its child nodes which did not have the highest propagation rating, prior to the receipt of the propagation signal just sent, and it would allow its (i.e., the formerly green node's) siblings to dock with it as child nodes).

If the propagation signal's recipient has other than the highest propagation rating (i.e., just prior to the receipt of the propagation signal), it would upgrade its propagation rating to the next highest rating grade; dock with its sibling which had the highest rating grade; and begin receiving content data from it. If the propagation signal's recipient has other than the highest propagation rating, it does not re-transmit the propagation signal to its own child nodes.

In the example illustrated in FIG. 23, nodes A and B receive the propagation signal from node P. Since node A has the highest propagation rating grade, here green, it: (i) sets its

reconfiguration flag buffer 136 (see FIG. 13) so that it has the lowest propagation rating grade, here red; and (ii) docks with node P (becoming a second level node 13) to begin receiving content data from node P. Node B changes its propagation rating from red to the next higher propagation rating (and since this example is a binary tree (or branch factor two) system, the next higher rating is the highest, green) and docks with node A to receive content data. The resulting topology is shown in FIG. 24. Note that node B remains a third level node 14.

FIGS. 25 and 26 illustrate what happens in this example when the departing node is green. In FIG. 25, node X is the departing node and it is green. When node P sends the upgrade propagation rating signal to node Q, it changes its propagation rating from red to green. In other respects the reconfiguration event proceeds essentially as described in the paragraph immediately above, and results in the topology shown in FIG. 26 (which is the same as the topology in FIG. 24).

FIG. 27 is a flow diagram associated with an example showing the Child Node's Propagation Routine, the routine which is followed by a child node upon receiving a propagation signal. Of note, this example is provided for illustration only, and is not restrictive, and other examples are of course possible (e.g., the color of a node may assigned by its parent such that nodes assign and know the color of their children but not of themselves).

In any case, still referring to FIG. 27, the child node first performs step 271 wherein it determines whether it has received such a signal. If it has not, then it does nothing. If it has received a propagation signal, it proceeds to step 272 wherein it checks its propagation rating grade in reconfiguration flag buffer 136 (see FIG. 13) . If its propagation rating is at the highest grade (i.e., it is a "green" node in this example), then it proceeds to step 273 where it sets its reconfiguration flag buffer to the lowest propagation rating grade. It then proceeds to step 274 in which it re-transmits the propagation signal to its own child nodes, and which results in the undocking of all its child nodes except for the one with the highest propagation rating. The child node (i.e., the child node referred to in the second sentence of this paragraph) then performs step 275 in which it determines whether it is already docked with the node which sent the propagation signal. If it is not (i.e., it received the propagation signal from its grandparent), then it proceeds to step 276. In that step it: (i) devises a new connection address list, which is the ancestor portion of its topology database with the current nominal parent node removed, resulting in the grandparent node becoming the first node on the connection address list; and (ii) then performs the Prospective Child Node's Connection

Routine (i.e., it goes to step 141 discussed above in connection with FIGS. 14 and 15). The Prospective Child Node's Connection Routine is performed because some likelihood exists that even the grandparent node may have departed from the distribution system between the moment it sent out the propagation signal and the moment that its highest rated grandchild from its missing child attempted to dock with it.

If the answer to the query in step 275 is in the affirmative (i.e., the child node is receiving a propagation signal which has been re-transmitted by its parent), then the child node does nothing more (or as illustrated in FIG. 27, it performs step 277 which is to remain docked to its parent node).

If the answer to the query in step 272 is in the negative (i.e., it does not have the highest propagation rating (it is a "red" node in this example)), then it proceeds to step 278 in which it: (i) sets its reconfiguration flag buffer so that its propagation rating is the next higher grade (in this binary tree system the rating is upgraded from red to green); (ii) undocks from the parent with which it was docked before receiving the propagation signal (if it was docked with a parent before receiving such signal); and (iii) either (a) waits a predetermined period of time during which the node's sibling which was green prior to their receipt of the propagation signal should have re-transmitted the propagation signal to its own child nodes (thereby causing any red child nodes to undock from it) or (b) confirms that the node's sibling which was green prior to their receipt of the propagation signal actual did re-transmit the propagation signal to its child nodes (if it has any). Then the child node performs step 279 in which it: (i) devises a new connection address list, which is the ancestor portion of its topology database with its sibling node having the highest propagation rating placed in front of the previous parent as the first node on the connection address list; and (ii) then performs the Prospective Child Node's Connection Routine (i.e., it goes to step 141 discussed above in connection with FIGS. 14 and 15). The Prospective Child Node's Connection Routine is performed because some likelihood exists that the sibling may have departed from the distribution system between the moment that the propagation signal had been sent to the child node and the moment that the child node attempted to dock with its sibling.

In another example (which example is intended to be illustrative and not restrictive), part (iii) of step 278 may be handled using the "priority join" mechanism. More particularly, in this example, a priority join from a node X cannot be refused by a target node T – it must accept (dock) the incoming node. If the target node T already had two children G & R, the target node T instructs its red child R to connect (via "priority join") to the green node G. The node T then disconnects from its red child R immediately prior to accepting the incoming

node X. The bumped node R then sends a priority join to its own green sibling A as the target. These actions cause the reconfiguration events to cascade until the edge of the tree is reached (of note, the algorithm described in this paragraph may have essentially the same effect as illustrated in Figures 28 and 29 – under certain circumstances the algorithm described in this paragraph may be somewhat more reliable).

FIG. 28 illustrates an example distribution system comprising node P as a first level node 12; nodes Q and X as second level nodes 13; nodes A and B as third level nodes 14; nodes C, D, E an F as fourth level nodes 15; nodes G and H as fifth level nodes 17; nodes I and J as sixth level nodes 20; nodes K and L as seventh level nodes 21 and nodes M and N as eighth level nodes 22. Nodes Q, B, C E, H, I, L and M have red propagation ratings, as symbolized by the dashed circles, and the remaining nodes have green propagation ratings as symbolized by the solid circles. As a result of node X's departing from the system, node P sends out an upgrade propagation rating signal to node Q and a propagation signal to the children of node P's departed child node X (i.e., nodes A and B).

In another example (which example is intended to be illustrative and not restrictive), during a graceful depart of node x, node x may send out messages to its parent and children immediately prior to its departure – rather than the parent of node x initiating these messages as described above.

In any case, continuing the example of FIG. 28, node Q changes its propagation rating to green. Further, in this example nodes A and B begin the Child Node's Propagation Routine. With respect to node A, it answers the query of step 271 in the affirmative. Since its rating is green, it also answers the query of step 272 in the affirmative. In step 273 node A changes the setting of its reconfiguration flag buffer to show a red propagation rating. In step 274 it re-transmits the propagation signal to its child nodes C and D. It answers the query of step 275 in the negative because it is not docked for purposes of receiving content data with node P. Node A then goes to step 276 wherein it consults the ancestor portion of its topology database and creates a new connection address list starting with its grandparent. The new connection address list is P-S. Then node A performs the Prospective Child Node's Connection Routine, starting with step 141. (See FIG. 14.) Assuming that no other intervening events have occurred, node A will successfully dock with node P.

Node B answers the query of step 271 in the affirmative and, because its propagation rating was red when it received the propagation signal, it answers the query of step 272 in the negative. From there it goes to step 278 in which it changes its propagation rating to green. Then in step 279 node B consults the ancestor portion of its topology database and creates a

new connection address list with its sibling, node A, placed in front of node X as the first address on the new connection address list. The new connection address list is A-X-P-S. Then node B performs the Prospective Child Node's Connection Routine, starting with step 141. Assuming that no other intervening events have occurred, node B will successfully dock with node A (if node A has disappeared, node B would attempt to dock with node X, but since it is not there, it would move on to node P.)

Node A, which had the higher propagation rating of the two child nodes of departed node X, moves up one level to become a second level node 13. Since it is a "new" entry to the second level, its initial utility rating and its propagation rating will be lower than that of node Q. As time goes by, at subsequent utility rating measurement events node A's utility rating (and hence its propagation rating) may become higher than that of node Q.

Node B, which had the lower propagation rating of the two child nodes of departed node X, does not re-transmit the propagation signal to its child nodes (nodes E and F), and they will follow B wherever it goes in the distribution system. In the example discussed here, node B becomes the child of node A while remaining third level node 14. At least initially, it will be node A's child with the higher utility rating and propagation rating.

As a result of node A's performing step 274, its child nodes C and D (which were fourth level nodes 15 in FIG. 28 when node X was still in the system) receive a propagation signal. Since node C has a red propagation rating, it, like node B, will remain in its level, change its propagation rating to green and dock with its sibling, with the result being that node D becomes its parent.

Since node D had a green propagation rating when it received the propagation signal, it answers the queries of steps 271 and 272 in the affirmative and changes its propagation rating to red in step 274. It answers the query of step 275 in the affirmative, and remains docked with node A. As a result, node D moves up a level and becomes an third level node 14 (with, at least until the next utility rating event, a lower utility rating and propagation rating than its new sibling, node B).

This process proceeds down the tree, with each child of a node which moves up a level doing one of the following under this example: (i) if it is a node having a green propagation rating, it remains docked with its parent, thereby itself moving up a level, changes its propagation rating to red and re-transmits the propagation signal to its child nodes; or (ii) if it is a node having a red propagation rating, it docks with the node which was its sibling, thereby staying in the same level, and changes its propagation rating to green.

The resulting topology after the reconfiguration event caused by node X's departure is illustrated in FIG. 29.

Of note, it is believed that reconfiguration using the Child Node's Propagation Routine of this example when a node departs the system does not necessarily result, in the long run, in significantly more inter-node reconnection events than any other reconfiguration method (and may result in fewer reconnection events). Moreover, it is believed that reconfiguration using the Child Node's Propagation Routine of this example helps assure that the more reliable nodes are promoted up the distribution network even if many reconfiguration events occur close in time to each other in a particular tree.

Referring now to a specific example of pseudo-code (which example is intended to be illustrative and not restrictive), it is noted that a propagation routine may be set forth as follows (wherein node X is the node departing from the distribution system):

```
Departs(X)
Begin
  If (X.Parent.GreenChild == X)  /* the departing node is green*/
  {
    X.Parent.GreenChild := X.Parent.RedChild
    X.Parent.RedChild := X.GreenChild
  }
  Else /* the departing node is already red */
  {
    X.Parent.RedChild :=X.GreenChild
  }
  Propagate(X.GreenChild, X.RedChild)
End
```

Referring now to another specific example of pseudo-code (which example is intended to be illustrative and not restrictive), it is noted that a propagation routine may be set forth as follows (wherein node promotion during the network reconfiguration event is precipitated by node X's departure):

```
Propagate (X, RedSibling)
Begin
  OriginalGreenChild := X.GreenChild
```

```
OriginalRedChild := X.RedChild
X.RedChild := X.GreenChild
X.GreenChild := RedSibling
If (OriginalGreenChild <> null)
{
  Propagate(OriginalGreenChild, OriginalRedChild)
}
End
```

Referring now to a "Malfunctioning Nodes" example (which example describes a voting system between grandchildren and grandparents about a potentially malfunctioning parent and is intended to be illustrative and not restrictive), it is noted that what happens when a node leaves the distribution network was discussed in the above example (e.g., when a node leaves the network, its parent will, at the next utility rating event, report that it has room for a new node, except when the parent has been contacted about the missing node by its grandchild node or nodes, in which event, a reconfiguration event will proceed as described above). Sometimes, however, a node intends to remain in the network, but there is a failure of communication.

In this regard, each node's topology database may include information regarding that node's ancestors and descendants. In the event that a child node stops receiving signals from its parent node, the child may, in one example, send a "complaint" message to its grandparent. The grandparent may check whether the parent is still there. If it is not, then the grandparent may send a propagation signal to the child nodes of the missing parent.

Further, if the grandparent detects that the parent node is actually still there, then the grandparent node may send a disconnect signal to the parent node (e.g., sending it back to the server to begin the connection process again). This may occur, for example, when one of the two following conditions is exists: (i) the child node is the only child node of the parent, or (ii) the child and its sibling(s) are complaining to the grandparent.

In this example, the grandparent would also send a propagation signal to the child nodes of the disconnected parent, and a reconfiguration event would occur.

However, if the child node has siblings and they are not sending complaint signals to the grandparent, then the grandparent may assume that the problem is with the complaining child node. Thus, grandparent may send a disconnect signal to the complaining child node (e.g., sending it back to the server to begin the connection process again). If the complaining

child node had its own child nodes, they would contact the departed child node's parent to complain, starting a reconfiguration event (in another example (which example is intended to be illustrative and not restrictive), the grandparent would send out signals to the malfunctioning parent (instructing it to depart) and to the children (instructing the green child to climb its path to its grandparent and instructing the red child to cross connect to the green child).

The foregoing example (which example is intended to be illustrative and not restrictive) can be described in connection with FIG. 30 (which depicts a tree in a binary system having a primary server 11, in which node A, a first level node 12, has two child nodes B and B' (which are second level nodes 13); node B has two child nodes C and C' (which are third level nodes 14); node C has two child nodes D and D' (which are fourth level nodes 15); and node D has two child nodes E and E' (which are fifth level nodes 17)) and FIG. 31 (which is a flow diagram showing an example Grandparent's Complaint Response Routine).

Who does node C complain to in this example when node C no longer gets satisfactory service from node B? Node C complains to its grandparent, node A. If node C does not hear back from node A within a predefined amount of time (e.g., if node A has left the network), node C will then exit the network and immediately reconnect as a new node by going to the primary server S for assignment.

What does node A do in response to a complaint from node C in this example? When node A receives a complaint from grandchild node C about node C's parent node B (i.e., node A's child node B), node A will choose to either remove its child node B or its grandchild node C. A will make this determination based on whether node C alone, or both node C and its sibling node C' are experiencing problems with node B, together with the knowledge of whether node A is continuing to receive satisfactory service reports from node B. If node A is continuing to get satisfactory service reports from node B and there is no indication that node C' (the sibling of node C) is experiencing problems with node B, then node A will assign the "fault" to node C and issue a disconnect order for its removal. At this point the green child of node C (i.e., node D or D') will move up a level connecting to parent B, and the red child of node C (the other of node D or D') will connect as a child of its former sibling. The reconfiguration event will then propagate as discussed above.

If, on the other hand, node A is not getting satisfactory service reports in this example from node B and/or a complaint from node C's sibling arrives within a `window` of node C's complaint, then node A will assign the "fault" to node B and issue a disconnect order for its

removal. At this point the green child of node B (i.e., node C or C') will move up a level connecting to grandparent node A, and the red child of node B (the other of node C or C'), will connect as a child of its former sibling. The reconfiguration event will then propagate as discussed above.

An exception to the above is the case where node C is the only child of node B. Under these circumstances node B may be disconnected in this example by node A based solely on the recommendation of node C. In another example, the system will not disconnect a node with one child based solely on the complaint of that child (in this case there may be insufficient data to accurately assign blame - disconnecting the node being complained against and promoting the complainer may have the potential to promote a node that is experiencing problems).

Referring now to FIG. 31 (showing a flowchart associated with this example), node A receives a complaint from node C about node B in step 311. Node A then goes to step 312 in which it checks its topology database to determine whether node C is the only child of node B. If the answer is no (as shown in FIG. 30), then node A goes to step 313 in which node A determines whether there is a communication problem between it and node B. If the answer is no, then node A proceeds to step 314 in which it determines whether it has received a similar complaint about node B from node C' within a predetermined period of time of node A's having received the complaint from node C. If the answer to that query is no, then node A goes to step 315 in which it sends a disconnect signal to node C. At that point node A has completed the Grandparent's Complaint Response Routine (in one example a propagation signal does not necessarily have to sent by node A to node C's child nodes - if any exist, they will complain to node B which, upon discerning that node C is no longer in the distribution network, will send a propagation signal to such complaining nodes).

If the response to any of the queries in steps 312, 313 and 314 is yes in this example, then node A proceeds to step 316 in which it disconnects node B. Node A then proceeds to step 317 in which it sends a propagation signal to its grandchild nodes (here nodes C and C') and a reconfiguration event will occur as described above.

The method of this example described above takes a "2 out of 3" approach on the question of node removal. In this regard, it is believed that many interior nodes in a binary tree distribution network constructed pursuant the present invention will have direct connections to three other nodes: their parent, their green child, and their red child. The connections between a node and its three "neighbor" nodes can be thought of as three separate communications channels. In this example a node is removed from the network

when there are indications of failure or inadequate performance on two of these three channels. When there are indications that two of the channels are working normally, the complaining node is presumed to be "unhealthy" and is removed.

In the case where an interior node has only a single child, the "complaint" of that child to its grandparent is sufficient, in this example, to remove the parent node - even when the node to be removed is communicating perfectly well with it's own parent. In other words, given the communication chain A-B-C where node A is the parent of node B and node B is the parent of node C, and given that node C has no siblings, then a complaint of node C to node A will cause node A to remove node B in this example regardless of the fact that node A and node B are not having communication problems (this course of action may help ensure that a potentially unreliable node does not move up the hierarchy of nodes, even at the cost of occasionally sending a "healthy" node back to the edge of the distribution chain based on uncorroborated complaints of its child). Of course, as mentioned above, there are examples where the system will not disconnect a node with one child based solely on the complaint of that child.

In another example of voting (which example is intended to be illustrative and not restrictive), instead of (or in conjunction with) nodes voting on the behavior of other nodes, nodes may continually grade their own performance and if they detect that there is insufficient bandwidth on two or more of their links (links to their parent and their children) the node will perform a graceful depart from the network (initiating a reconfiguration as described elsewhere) and then the node will either climb its path to an ancestor node (or go directly to the server) to request a connection path to the edge of the tree. This system allows nodes to "step aside" when they detect connection / bandwidth related issues.

Referring now to a "Stepped Delay of Playing Content Data" example (which example is intended to be illustrative and not restrictive) it is noted that in order to have all the user nodes experience the playing of the content data at approximately the same time, regardless of the level in which the node resides, the content data buffer 125 shown in FIG. 13 of a user node may be larger the higher uptree it is in a distribution chain (i.e., the closer it is to the root server). The larger the buffer, the greater the amount of time between a particular event (or from the node's receiving the content data), and the node's actually playing the content data in its content data buffer. To the extent that the content data buffer of a node is sized to vary the time that the playing of the content data is started, the content data buffer is a delay. Alternatively, the delay may be a timer or a delay buffer 126 as shown in

FIG. 13, or a combination of elements. The delay's purpose may be to help assure that all users experience the play of the content data approximately simultaneously.

For a distribution network having n levels of user nodes cascadingly connected to a server node, where n is a number greater than one (and therefore the distribution network comprises the server node and first through nth level user nodes), the period of time (i.e., the delay time) between:

(i) the moment content data is received by a node or, from a predetermined moment such as, for example, a reporting event or utility rating event (the occurrence of which may be based on time periods as measured by a clock outside of the distribution system and would occur approximately simultaneously for all nodes); and

(ii) the playing of the content data by the node;

is greater for a node the higher uptree the node is on the distribution chain. That is, the delays in first level nodes create greater delay times than do the delays in second level nodes, the delays in second level nodes create greater delay times than the delays in third level nodes and so on. Described mathematically, in an n level system, where x is a number from and including 2 to and including n, the delay time created by a delay in an (x-1) level node is greater than the delay time created by a delay in an x level node.

By way of example (which example is intended to be illustrative and not restrictive), first level nodes could have a built in effective delay time in playing content data of one-hundred-twenty seconds, second level nodes could have a built in effective delay time of one-hundred-five seconds, third level nodes could have a built in effective delay time of ninety seconds, fourth level nodes could have a built in effective delay time of seventy-five seconds, fifth level nodes could have a built in effective delay time of sixty seconds, sixth level nodes could have a built in effective delay time of forty-five seconds, seventh level nodes could have a built in effective delay time of thirty seconds, and eighth level nodes could have a built in effective delay time of fifteen seconds.

The delays may take advantage of the fact that transmission of the packets forming the content data from one node to another may take less time than the actual playing of such packets. This may allow for essentially simultaneous playing of the content data throughout all the levels of the distribution network. The delays may also allow nodes to handle reconnection and reconfiguration events without any significant loss of content data.

In another example (which example is intended to be illustrative and not restrictive), the system may not endeavor to have buffers closer to the server contain more data than buffers of nodes that are further away. Instead, every node may have a buffer that can hold,

for example, approximately two minutes worth of A/V content. The system may endeavor to fill every node's buffer as rapidly as possible, regardless of the node's distance from the server, in order to give each node the maximum amount of time possible to deal with network reconfiguration and uptree node failure events.

Of note, nodes may receive a "current play time index" when they connect to the server. This value may be used to keep all nodes at approximately the same point in the stream – to have all nodes playing approximately the same moment of video at the same time – by telling them, essentially, begin playing A/V content at this time index. The play time index may be, for example, approximately two minutes behind "real time" – allowing for nodes to buffer locally up to two minutes of content (the present invention may permit drift between nodes within the buffer (e.g., two minute buffer) or the present invention may reduce or eliminate this drift to keep all nodes playing closer to "lock step").

Turning now to a general description of an example process for Node X to enter a network including server Node S, Node X must contact Node S to initiate the process. For illustrative purposes, in the following example (which example is intended to be illustrative and not restrictive) each node in the network may have no more than two connections, and the root server (i.e., the highest node in the tree) shall be referred to as Node S.

The process of this example begins with Node X making a request of Node S to join the network.

If Node S has an open connection (e.g., including one or more connected bumpable non-repeat-capable nodes which may be replaced due to an incoming connection request from a repeat-capable node), Node S can accept Node X's request to join the network, and Node X becomes a child of Node S.

If, on the other hand, Node S does not have an open connection, Node S may reject Node X's request, and may provide to Node X a node-by-node path from Node S to Node P, which Node S understands to be in the network, and under which, as determined by Node S, Node X may be able to join the network.

The process of this example continues with Node X making a request of Node P to join the network.

If Node P does not respond – regardless of why Node P does not respond – Node X may request connection from the next higher node in the path. For the purpose of this illustration, thereafter the next higher node is now considered Node P. Node X may make such a request of successive Node Ps, until Node X receives a response.

If a particular Node P has an open connection, that Node P can accept Node X's request to join the network, and Node X becomes a child of that particular Node P.

If, on the other hand, a particular Node P does not have an open connection, that particular Node P may reject Node X's request, and may provide to Node X a node-by-node path from that particular Node P to Node P', which the instructing Node P understands to be in the network, and under which, as determined by the instructing Node P, Node X may be able to join the network.

If Node X receives a further rejection, Node X may then treat the Node P' as a Node P and continue the process.

In general, under this example the process is continued until Node X finds a connection.

Referring now more specifically to a "Registration Process" example involving two arbitrary Nodes X and N (neither of which is necessarily a server), FIG. 32 shows Node X sending a request to Node N for acceptance into the distribution network tree (of which Node N is part).

In this example, Node N (which may be thought of as an instructing node) may generally return one of three codes (or messages): Accept, Path, or Reject. FIGS. 33A-33C show example messaging between Node X and Node N under this example. More particularly:

- FIG. 33A shows Node X sending a Register message to Node N and Node N responding to the Register message with an Accept message
- FIG. 33B shows Node X sending a Register message to Node N and Node N responding to the Register message with a Path message (e.g., identifying a path from Node N to a new instructing node)
- FIG. 33C shows Node X sending a Register message to Node N and Node N responding to the Register message with a Reject message (i.e., rejecting Node X's request for a connection to Node N).

In another example, the registration process may be extended to include some system maintenance or configuration activities. Such maintenance or configuration activities may include "flood testing" (discussed in detail below), identification of a bad protocol, and/or Firewall Status (discussed in detail below).

In this regard, FIG. 34A shows the messaging (in connection with registration with a flood test) between Node X, a node requesting to join the network, Node S, a server node

within the network, and Node R (e.g., a random (or predetermined) node that may, but not need be, within the network).

Further, FIG. 34B shows the messaging (in connection with registration with a bad protocol version) between Node X and Node S, a server node within the network. Of note, such a bad protocol message may be sent by the server to a connection requesting node to inform that node that it is running an obsolete version of the client software and must upgrade to the current version before being allowed to join the network.

Further still, FIG. 34C shows the messaging (in connection with registration with standard response) between Node X and Node S, a server node within the network (wherein such standard response includes, for example, Accept / Reject / Path).

Referring now in more detail to the "flood test" example shown in FIG. 34A, this example process begins, as above, with Node X making a request to Node S to join the network. Node S then sends Node X a Meta message (such a Meta message may be sent by the server to a connection requesting node to inform that node of certain characteristics of the network such as, for example: the streaming bandwidth rate used by the network, the server's public key for authenticating that secure packets originated from this server, and any other information the architects of the network might deem necessary or desirable for nodes to know when connecting to the network). Node S may also require that Node X undergo performance or flood testing before being accepted or placed within the network. In such an event, Node S responds to Node X's request to join the network with an instruction that Node X undergo a flood test (e.g., of Node X's upstream channel). Such an instruction may be accompanied by an address of Node R, the illustrative flood test target. Note that any node within or without the network may conduct the flood test. One purpose of the flood test may be to permit Node S to be aware of Node X's upstream capability before determining where (or if) to assign Node X in the network.

Where Node S requires that Node X undergo performance or flood testing before being accepted or placed within the network, Node S provides a message so informing Node X, shown as "DoFlood" in FIG 34A. The DoFlood message may provide to Node X the address of the flood test target, but the flood test target could be Node S itself, or a predetermined flood test target.

In response to the DoFlood message, Node X sends an InitFlood message to the flood test target. In one embodiment, Node X would await an acknowledgement (not shown) of its InitFlood message, but in another embodiment, awaiting an acknowledgement is not

necessary. Node X would then begin transmitting FloodData messages, as shown in FIG 34A.

Since, in this example, FloodData messages are transmitted for a purpose of determining Node X's upstream bandwidth, in one embodiment, Node X would transmit an amount of FloodData to the flood test target as shown in FIG 34A. In various embodiments, the maximum amount of FloodData may be limited to a fixed number of FloodData messages, to a fixed amount of data transmitted within FloodData messages, or to transmission of FloodData or FloodData messages over a fixed period of time.

To protect the flood test target from being inundated with FloodData (e.g., where Node X has a very high upstream bandwidth capability), Node X may cap (e.g., be so directed by Node S) the FloodData transmission to, e.g., 100 Kbits/sec, 200 Kbits/sec, or some other data rate that would be a desired data transmission rate for a node on the particular network of interest.

Thus, by receiving the FloodData messages, the flood test target can determine, for example, an upstream capability rating of Node X (provided that the flood test target has the same or more bandwidth in the downstream channel compared to the upstream capability of Node X.

In another example, both the target and the sender will know the bandwidth of the sender at the conclusion of the flood test. More particularly, the present invention may use TCP/IP (rather than UDP), and all packets sent may receive an acknowledgement. In this example, Node X, the node sending out the flood packets, sends out the next flood packet only after receipt of the first packet has been acknowledged by the target of the flood test. Thus, the number of flood data packets Node X is able to send per unit time (along with the size of those packets) may allow Node X to calculate its upstream bandwidth.

In another example, the target of the "flood test" may be any node including the server but excluding any firewalled or low-bandwidth nodes.

After the "flood test", Node X may then connect with Node S (or be instructed to go to another node if Node S can not/will not accept Node X). Of note, if Node X fails the "flood test", Node X may be sent far downtree or may be refused entry into the network entirely. Further, if node N is accepted under these circumstances it will be deemed a "bumpable" non-repeat-capable node.

Referring now in more detail to the "Firewall Status" example shown in FIG. 34C, this example process begins, as above, with Node X making a request to Node S to join the network. Node S then send Node X a Meta message. A firewall status message is a message

sent over a new connection from (in the case of Figure 34C) Node S to Node X. If Node X receives this message, over a communications channel initiated by Node S, Node X will know that it is not firewalled. Failure to receive a firewall status message when a node is expecting such a message allows that node to conclude that it is in fact firewalled. Thus, if Node X is expecting to receive a firewall status message over new connection initiated by some other node and that message does not arrive, Node X can conclude (with a high degree of likelihood of being correct) that it is firewalled. Also, since (in one example) the present invention uses the TCP/IP messaging protocol the sender, Node S, will also know whether the firewall status message was successfully received by Node X and thus Node S will also know whether Node X is or is not firewalled (one or more other nodes may also receive data regarding the firewall status).

After receiving the firewall status report (not shown) Node S will then respond with a standard response (accept, path, reject) as discussed above.

To now give a specific example of the use of both the "flood test" and the "firewall status" mechanisms during registration of a new node, FIG. 35A shows a state of the network before Node X joins the network and FIG. 35B shows a state of the network after Node X joins the network. Further, FIG. 36 shows the messaging among Node X, Node S, Node B and Node R. More particularly:

- Node X sends a Register message to Node S and Node S responds with a Meta message (providing Node X with basic information about the network) and a DoFlood message (which requests Node X to determine its bandwidth and report back to Node S when Node X has that information). Node S knows to send a "DoFlood" to Node X because the initial Register message sent by Node X to Node S contains an indication that Node X does not yet know its current upstream bandwidth capacity. The DoFlood message will contain the address of a target node (Node R), selected by Node S, that Node X should flood against.
- Node X sends an InitFlood message to Node R and then "floods" Node R with data for some period of time (e.g., 15 seconds). At the conclusion of this test Node X will be able to compute its upstream bandwidth based on the number and size of data packets it was able to successfully transmit to Node R.

- Node X sends another Register message to Node S and Node S responds with a Meta message. Contained within this second Register message is the result of Node X's "flood test" which gives an indication of Node X's upstream bandwidth capacity.
- Node S now tests whether external nodes can initiate communication with Node X by sending Node X a Firewall Status message over a new connection. If this message is successfully received by Node X (and acknowledgement received by Node S), both Nodes X and S will know that Node X is not firewalled (on a network communication port).
- Node S sends a Path message to Node X. This message will include a list of nodes that X should attempt connection with, beginning with the preferred target node (e.g., Node B). As described elsewhere in the application, the choice of target node, and hence the contents of the path, may be influenced by both the current configuration of the network and whether or not Node X is repeat-capable (based on the results of the "flood test" and "firewall test").
- Node X sends a Register message to Node B and Node B responds with an Accept message.

Referring now to a "Firewall Request / Firewall Status" example involving two arbitrary Nodes A and B (neither of which is necessarily a server), FIG. 37 shows a portion of a network including Node A and Node B. In this example (as shown in FIG. 38), a Firewall Request message is sent by Node B to request that the recipient of the message, Node A, test Node B's Firewall Status. The receiving Node A may send (e.g., over a new connection), the Firewall Status message to Node B. If Node B timely receives and interprets the Firewall Status message it can be certain that it is a non-firewalled node. If Node B does not timely receive a Firewall Status message from Node A then Node B may assume that it is firewalled.

In another example (which example is intended to be illustrative and not restrictive), if Node A fails to connect to Node B or to send the Firewall Status message no further action is needed on the part of Node A.

Node B may also augment its assumption about its Firewall Status by using a three strikes (i.e., three attempts) method before considering itself to be firewalled (of course, three strikes is provided as an example only, and any desired number of "strikes" or attempts may be used).

In another example (which example is intended to be illustrative and not restrictive), the multiple attempts may be carried out within a predetermined time "window" (after which receiving no Firewall Status message Node B may consider itself firewalled).

In another example (which example is intended to be illustrative and not restrictive), Firewall Requests may be sent out by all nodes (e.g., at regular time intervals) in order to maintain a current and accurate picture of their Firewall Status.

In another example (which example is intended to be illustrative and not restrictive), Firewall Requests may be sent out by certain node(s) (e.g., at regular time intervals) in order to maintain a current and accurate picture of their Firewall Status.

In another example (which example is intended to be illustrative and not restrictive), the Firewall Status message may be sent by the server on initial registration (e.g., without the need for a Firewall Request message and both the sender (the server) and the receiving node (the node that is registering) may use the successful delivery of the Firewall Status message as an indicator of the connecting node's Firewall Status.

Referring now to a "Propagate Path / Down" example (which example is intended to be illustrative and not restrictive), FIG. 39A shows a portion of a network including Node A and Node B. Further, FIG. 39B shows an example of nodes in a tree sending Propagate Path / Down messages downtree to their children (e.g., at regular time intervals).

In one example (which example is intended to be illustrative and not restrictive), the Propagate Path / Down message may include topology information containing a path back to the root server and the time interval for sending such message may be calculated on a node-by-node and/or level-by-level basis relative to the amount of data to be sent in the Propagate Path / Down message (wherein the amount of data in the Propagate Path / Down message is related to a particular node's (or level's) downtree position relative to the root server). Note, that although this time interval may not be very important for Propagate Path / Down, since the amount of data for each successive downtree node (or level) may increase relatively slowly (e.g., linearly), the time period may be of critical importance for Propagate NTM / Up (which is discussed below), since the amount of data for each successive uptree node (or level) may increase relatively quickly (e.g., exponentially). Of note, the acronym "NTM" refers to the Network Topology Model.

In any case, FIG. 40 shows the messaging between Node A and Node B in connection with a Propagate Path / Down message (i.e., Node A sends a Propagate Path message downtree to Node B).

Referring now to a "Propagate NTM / Up" example (which example is intended to be illustrative and not restrictive), FIG. 41A shows a portion of a network including Node A and Node B. Further, FIG. 41B shows an example of nodes in a tree sending Propagate NTM / Up messages uptree to their parents (e.g., at regular time intervals).

In one example (which example is intended to be illustrative and not restrictive), the Propagate NTM / Up message may include information containing the topology of the subtree(s) below a particular node and the time interval for sending such message may be calculated on a node-by-node and/or level-by-level basis relative to the amount of data being sent in the Propagate NTM / Up message (wherein the amount of data in the Propagate NTM / Up message is related to a particular node's (or level's) uptree position relative to tree height (e.g., the height of a particular node above its lowest leaf level)). Note that in this example (which example is intended to be illustrative and not restrictive), the NTM data that the node sends up is rooted at itself and contains both its children.

In any case, FIG. 42 shows the messaging between Node A and Node B in connection with a Propagate NTM / Up message (i.e., Node B sends a Propagate NTM message uptree to Node A).

In another embodiment, the following process may be employed when a node enters the network. More particularly, when the server adds a node X to the system (gives it a connection path to node Y) the server will add node X to its network topology model as a child of node Y. In other words, the server will assume that the connection will take place normally with no errors. The reason why doing so may be critical to the functioning of the system in this embodiment is that if the server failed to account for the fact that node X is now (probably) a child of node Y, using the "keep the tree balanced" approach embodied in the Universal Connection Address Routine, all new nodes connecting to the system would be assigned as children of node Y up until the point where a propagate up arrives indicating that node X was successfully added to the tree. If the tree is large and node Y is many levels removed from the server it could be some time (e.g., a minute or more, in very large trees several minutes) before confirmation arrives that node X truly connected to node Y as its child.

Now, in addition to simply placing node X into the server's NTM (Network Topology Model) and being done with the matter, in this embodiment the system must account for the fact that it may take a number of propagate NTM/up cycles for the true status of node X to be reported back to the server. Thus, for some number of propagate cycles (or some amount of time) the server may compare the newly propagated (updated) NTM it is receiving from its

children to the list of "recently added nodes". If a recently added node is not in the propagated NTM it will be re-added to the NTM as a child of node Y. At some point node X will no longer be considered "recently added" and the server will cease adding node X back into its NTM. In other words, the server will assume that, for whatever reason, node X never successfully connected to the network.

In another embodiment, the following process may be employed when a node gracefully exits the system. More particularly, when a node (node X) leaves the network by performing a graceful exit (e.g., the user selects "exit" or clicks the "x" or issues some other command to close the application) the node will send signals to its parent and child nodes (if any) immediately prior to actually shutting down. Upon receipt of such a "depart" message, the parent of the departing node (node P) will modify its internal NTM (Network Topology Model) to a configuration that corresponds to the predicted shape of the network after the departing node exits. In this embodiment, node P will assume that its NTM is accurate and that the entire cascade of reconfiguration events initiated by the departure of node X will happen normally. As such, node P will now expect the former green child of node X (node XG -- if such a child exists) to connect – essentially taking node X's place in the network.

At this point, node P's NTM represents a prediction of the (anticipated) state of the network rather than the actual reported configuration. In order to indicate that node P does not (yet) have a physical connection to node XG, node P will mark node XG as "soft" in its NTM – meaning node XG is expected to take up this position in the network but has not yet physically done so. Node P will report this "predicted" NTM (along with node XG's status as a soft node) up during its propagate NTM/Up cycles, until node P physically accepts a child connection in node XG's "reserved" spot and receives an actual report of the shape of the downtree NTM from that child, via propagate NTM/ups.

Of note, the above actions may be taken in order to have the propagated NTM's reflect, as accurately as possible, the topology of the network.

Of further note, because of the distributed nature of a network according to the present invention, a complete and accurate picture of the current shape of the network at any instance in time can never be guaranteed. In fact, though the root server has the "widest" view of the network, without the heuristics described herein, that view would be the "most dated" of any node in the network. Thus, heuristics such as those just described may be vital to producing NTM's that are as close to complete and accurate as possible (e.g., so that the Universal Connection Address Routine can position incoming nodes so as to keep the distribution tree as nearly balanced as possible).

Referring now to a "Banned List" example (which example is intended to be illustrative and not restrictive), it is noted that such a "banned list" (or "banished list") may be used for keeping certain nodes off of the network, removing certain nodes from the network, and/or keeping certain nodes as far downtree as possible (e.g., in the case of a binary tree network keeping certain nodes as "red" nodes).

More particularly, FIG. 43 shows a portion of a network including Node A, Node B, Node C and Node D and FIG. 44 shows the messaging between Node A, Node B, Node C and Node D in connection with Banned List messages (i.e., in this FIG. 44, Node A sends a Banned List message to Node B and Node B sends a Banned List message to Nodes C and D).

In one example (which example is intended to be illustrative and not restrictive), nodes receiving this message should re-transmit the message downtree to all children in an unaltered fashion.

In another example (which example is intended to be illustrative and not restrictive), the Banned List messages may include IP lists and/or URL's.

In another example (which example is intended to be illustrative and not restrictive), the Banned List messages may originate at the server.

In another example (which example is intended to be illustrative and not restrictive), Banned List messages may be sent and/or originated by any desired node(s).

In another example (which example is intended to be illustrative and not restrictive), Banned List messages and may be sent using a Secure Packet format.

In another example (which example is intended to be illustrative and not restrictive), node(s) may store the content of the message (e.g., the banned list itself) for future use.

In another example (which example is intended to be illustrative and not restrictive), the server (or any other desired node(s)) may send out and/or originate the Banned List messages at regular time intervals.

Referring now to a "Bumpable Node" embodiment, it is noted that this "bumpable node" mechanism may be used to "bump" (or send downtree) a particular node (e.g., when a new node is added). More particularly, in one example (which example is intended to be illustrative and not restrictive), a server may instruct Node X to connect with Node P and Node P may know to "bump" its child, Node C, by: (a) instructing Node C to disconnect from Node P; (b) allowing Node X to connect to Node P; and (c) instructing Node C to re-connect as a child of Node X. Of note, such a "bumping" operation may be carried out, for example (which example is intended to be illustrative and not restrictive), because the

bumped node is firewalled and/or because the bumped node has failed an upstream and/or downstream bandwidth test (which test may be carried out by the server or by any other node(s)).

Of note, as used in this application, the terms "bumpable nodes" or "bumpable non-repeat-capable nodes" or "non-repeat-capable nodes" are nodes which either: (a) have insufficient bandwidth to act as repeaters in the network or (b) are firewalled.

Of further note, in one example the more inclusive "is bumpable" test may replace the "is low bandwidth" test in the Universal Connection Address Routine of Figure 20.

Referring now to a "Load Manager" embodiment, it is noted that this "load manager" mechanism may be used to rate limit (i.e., slow or stop) the addition of new nodes to the network. Such rate limiting may be useful because the larger the network grows, the more nodes can join per unit of time.

In one example (which example is intended to be illustrative and not restrictive), the rate limited addition of new nodes may be carried out by a server (or connection server) and may be based upon the length of time that it takes for the buffers associated with one or more nodes (or levels) of the existing network to fill (or reach a certain value). In a more specific example, the rate limited addition of new nodes may be based upon the length of time that it takes for the buffers associated with potential parents of the new nodes to fill (or reach a certain value).

In another example (which example is intended to be illustrative and not restrictive), the rate limited addition of new nodes may be carried out by any desired node(s) (e.g., one or more user nodes and/or a combination of one or more user nodes and a server (or connection server) node).

Referring now to a "Node Interlacing" embodiment, it is noted that this "node interlacing" mechanism may be used to add new nodes to the network by interlacing the new nodes on a level-by-level basis as shown in FIG. 45.

More particularly, as seen in this FIG. 45, new nodes may be added in the order indicated (as can be seen below, this example (which example is intended to be illustrative and not restrictive) relates to a binary tree network). That is, Node 1 may first be added as a child of the server followed by Node 2 as a child of the server – this will fully populate the first level; Node 3 may next be added as a child of Node 1, followed by Node 4 as a child of Node 2, followed by Node 5 as a child of Node 1, followed by Node 6 as a child of Node 2 – this will fully populate the second level; Node 7 may next be added as a child of Node 3, followed by Node 8 as a child of Node 5, followed by Node 9 as a child of Node 4, followed

by Node 10 as a child of Node 6, followed by Node 11 as a child of Node 3, followed by Node 12 as a child of Node 5, followed by Node 13 as a child of Node 4, followed by Node 14 as a child of Node 6 – this will fully populate the third level, etc.

Of note, such node interlacing may be used to efficiently fill (or bring to a certain value) the buffers associated with one or more nodes (or levels) of the existing network while new nodes are added.

Of further note, such node interlacing may help prevent adding a second child to a parent until the first child's buffer is full (or brought to a certain value).

Of still further note, such node interlacing may help to: (a) prevent taxing the parent's downtree channel; and/or (b) prevent two children being left with partially filled buffers in the event that their parent departs.

Of still further note, such node interlacing may be used, for example, in conjunction with or as an alternative to the load balancing mechanism discussed above.

In another embodiment the present invention may be used when a plurality of computer systems are connected (e.g., to the Internet) by one or more of an ISDN line, a DSL (e.g., ADSL) line, a cable modem, a T1 line and/or an even higher capacity link.

In another embodiment a distribution network for the distribution of data from a server node to a plurality of user nodes connected to the server node, wherein each of the user nodes has associated therewith a buffer is provided, which distribution network comprises: a mechanism for rate limiting the addition of new user nodes to the distribution network, wherein the rate limited addition of new user nodes to the distribution network is based at least in part upon the length of time that it takes for at least one buffer associated with at least one user node to reach a predetermined percentage of capacity, wherein the predetermined percentage of capacity is greater than zero.

In another embodiment a distribution network for the distribution of data from a server node to a plurality of user nodes connected to the server node, wherein each of the user nodes is in one of a plurality of levels located downtree from the server node is provided, which distribution network comprises: a mechanism for adding new user nodes to the distribution network in an interlaced manner, wherein the interlacing provides that each sibling user node in a given level of the distribution network is associated with just one child before any siblings in the given level are associated with two children.

The following embodiments will now describe a number of additional example algorithms which may be utilized in whole or in part to implement the present invention:

In one embodiment a process for docking a connection requesting user node with a distribution network is provided, said process including the following steps:

(a) having an instructing node receive a connection request from said connection requesting user node;

(b) forming an instructing node topology database indicating, at a point in time:

(i) which, if any, user nodes are docked with said instructing node as child nodes; which, if any, user nodes are docked with each of said child nodes as grandchild nodes of said instructing node; which, if any, user nodes are docked with each of said grandchild nodes as great-grandchild nodes of said server node; and so on; and

(ii) said docked user nodes respective bandwidth capacities;

(c) selecting from a group of nodes including the instructing node and said docked user nodes a recommended parent node for said connection requesting user node, wherein said recommended parent node has apparent available capacity to transmit content data to said connection requesting user node and is at least as close (i.e., in terms of network topology) to said instructing node as any other docked user node having apparent available capacity to transmit content data to said connection requesting user node (of note, placement may also or alternatively depend, as described elsewhere in the present application, in whole or in part on whether the connection requesting node is repeat-capable or not (i.e., whether it can support children)); and

(d) providing a connection address list to said connection requesting user node, said connection address list listing each node in said instructing node's topology database from said recommended parent node back to said instructing node.

In one example, the process may further comprise the following steps:

(e) having said connection requesting user node go to (or attempt to go to) the node at the top of said connection address list;

(f) having said connection requesting user node determine whether the node at the top of said connection address list is part of the distribution network;

(g) if the node at top of said connection address list is not part of the distribution network, deleting such node from said connection address list and repeating steps (e) and (f) with respect to the next node at the top of said connection address list;

(h) if the node at the top of said connection address list is part of said distribution network, having said connection requesting user node dock with said node at the top of said connection address list.

In another example, step (d) may further comprise the following step:

(i) if said instructing node is not a server node, including in said connection address list a list of nodes cascadingly connected with each other from the instructing node back to said server node.

In another example, the process may further comprise the following steps:

(e) having said connection requesting user node go to (or attempt to go to) the node at the top of said connection address list;

(f) having said connection requesting user node determine whether the node at the top of said connection address list is part of the distribution network;

(g) if the node at the top of said connection address list is not part of the distribution network, deleting such node from said connection address list and repeating steps (e) and (f) with respect to the next node at the top of said connection address list;

(h) if the node at the top of said connection address list is part of said distribution network, determining whether said node at the top of said connection address list has available capacity to transmit content data to said connection requesting user node; and

(i) if the node at the top of said connection address list is part of said distribution network and has available capacity to transmit content data to said connection requesting user node, having said connection requesting user node dock with the node at the top of said connection address list.

In another example, the process may further comprise the following steps:

(j) if the node at the top of connection address list is part of said distribution network and said node at the top of said connection address list does not have available capacity to transmit content data to said connection requesting user node, repeating steps (a)-(d), with said node at the top of connection address list being said instructing node and a new connection address list being the product of steps (a)-(d).

In another example, the process may further comprise the steps of counting the times that step (j) is performed and, when step (j) is performed a predetermined number of times, having said connection requesting user node repeat steps (a)-(d) with said primary server being the instructing node.

In another embodiment a process for docking a connection requesting user node with a distribution network is provided, said process including the following steps:

(a) having an instructing node receive a connection request from said connection requesting user node;

(b) forming an instructing node topology database indicating, at a point in time:

(i) which, if any, user nodes are docked with said instructing node as child nodes; which, if any, user nodes are docked with each of said child nodes as grandchild nodes of said instructing node; which, if any, user nodes are docked with each of said grandchild nodes as great-grandchild nodes of said server node; and so on; and

(ii) said docked user nodes' respective bandwidth capacities;

(c) assigning utility ratings to each of the docked user nodes, said utility ratings being a function of at least the bandwidth capacity of the docked user node and the elapsed time that the docked user node has been docked with the distribution network;

(d) selecting from a group of nodes including the instructing node and said docked user nodes a recommended parent node for said connection requesting user node, wherein said recommended parent node has apparent available capacity to transmit content data to said connection requesting user node and is at least as close (i.e., in terms of network topology) to said instructing node as any other docked user node having apparent available capacity to transmit content data to said connection requesting user node, and when the closest (i.e., in terms of network topology) docked user node with apparent available capacity to transmit content data to said connection requesting user node is equidistant (i.e., in terms of network topology) to said instructing node with at least one other docked user node with apparent available capacity to transmit content data to said connection requesting user node, said recommended parent node is the docked user node having the highest utility rating among such equidistant (i.e., in terms of network topology) docked user nodes with apparent available capacity; and

(e) providing a connection address list to said connection requesting user node, said connection address list listing each node in said instructing node's topology database from said recommended parent node back to said instructing node and, if said instructing node is not a primary server node, back to said primary server node.

In another embodiment a process for docking a connection requesting user node with a distribution network is provided, said process including the following steps:

(a) having an instructing node receive a connection request from said connection requesting user node;

(b) forming an instructing node topology database indicating, at a point in time:

(i) which, if any, user nodes are docked with said instructing node as child nodes; which, if any, user nodes are docked with each of said child nodes as grandchild nodes of said instructing node; which, if any, user nodes are docked with

each of said grandchild nodes as great-grandchild nodes of said instructing node; and so on; and

(ii) said docked user nodes' respective bandwidth capacities, including a designation of whether said respective docked user node's bandwidth capacity is below a predetermined threshold (e.g., said respective docked user node is a low-bandwidth node) or at least said predetermined threshold (e.g., said respective docked user node is a high-bandwidth node);

(c) forming a primary recommended parent node list ("PRPL") comprised of the instructing node (if it has available capacity to transmit content data to said connection requesting user node) plus those docked user nodes having apparent available capacity to transmit content data to said connection requesting user node, with said instructing node being placed first on the PRPL (if it is on the PRPL) and said docked user nodes having apparent available capacity to transmit content date to said connection requesting user node being ranked with those docked nodes which are closer (i.e., in terms of network topology) to the instructing node being ranked higher than those docked nodes which are further away (i.e., in terms of network topology) and with equidistant (i.e., in terms of network topology) docked user nodes being ranked such that those docked nodes which have the higher utility ratings being ranked higher than docked nodes having lower utility ratings;

(d) forming a secondary recommended parent node list ("SRPL") comprised of the instructing node (if it has no available capacity to transmit content data to said connection requesting user node but does have at least one low-bandwidth node docked directly to it) plus said docked user nodes which (i) are high-bandwidth nodes with no available capacity to transmit content data to said connection requesting user node and (ii) have at least one low-bandwidth node docked directly to it, with said instructing node being placed first on the SRPL (if it is on the SRPL) and said docked user nodes on the SRPL being ranked with those docked nodes which are closer (i.e., in terms of network topology) to the instructing node being ranked higher than those docked nodes which are further away (i.e., in terms of network topology) and with equidistant (i.e., in terms of network topology) docked user nodes being ranked such that those docked nodes which have the higher utility ratings being ranked higher than docked nodes having lower utility ratings;

(e) determining whether the connection requesting user node is a low-bandwidth node;

(f) if the connection requesting user node is a low-bandwidth node:

(i) selecting the highest ranked node on the PRPL as a recommended parent node; and

(ii) providing a connection address list to said connection requesting user node, said connection address list listing each node in said instructing node's topology database from said recommended parent node back to said instructing node and, if said instructing node is not a primary server node, back to said primary server node; and

(g) if the connection requesting user node is a high-bandwidth node:

(i) selecting the highest ranked node on the PRPL as a recommended parent node;

(ii) selecting the highest ranked node on the SRPL as an alternate recommended parent node;

(iii) determining whether the alternate recommended parent node is closer (i.e., in terms of network topology) to the server node than the recommended parent node;

(iv) if the alternate recommended parent node is not closer (i.e., in terms of network topology) to the server node than the recommended parent node, providing a connection address list to said connection requesting user node, said connection address list listing each node in said instructing node's topology database from said recommended parent node back to said instructing node and, if said instructing node is not the primary server node, back to said primary server node;

(v) if the alternate recommended parent node is closer (i.e., in terms of network topology) to the server node than the recommended parent node, (1) forcing the disconnection of a low-bandwidth node from the alternate recommended parent node and (2) providing a connection address list to said connection requesting user node, said connection address list listing each node in said instructing node's topology database from said alternate recommended parent node back to said instructing node and, if said instructing node is not the primary server node, back to said primary server node.

In one example, step (f)(ii) may include the following steps:

(1) adding said connection requesting user node to the topology database as a child of the recommended parent node; and

(2) if said recommended parent node would have no apparent available capacity to transmit content data to an additional node with said connection requesting user node docked with it, deleting the recommended parent node from the PRPL and adding it to the SRPL.

In another example, step (g)(iv) may include the following steps:

(1) adding said connection requesting user node to the topology database as a child of the recommended parent node;

(2) if said recommended parent node would have no apparent available capacity to transmit content data to an additional node with said connection requesting user node docked with it, deleting the recommended parent node from the PRPL; and

(3) if said recommended parent node is deleted from the PRPL but has at least one low-bandwidth node docked directly to it, adding the recommended parent node to the SRPL.

In another example, step (g)(v) may include the following steps:

(3) adding said connection requesting user node to the topology database as a child of the alternate recommended parent node; and

4) if said alternate recommended parent node would have no apparent available capacity to transmit content data to an additional node with said connection requesting user node docked with it, deleting the alternate recommended parent node from the SRPL.

In another example, the process may further comprise the following steps:

(h) having said connection requesting user node go to (or attempt to go to) the node at the top of said connection address list;

(i) having said connection requesting user node determine whether the node at the top of said connection address list is part of the distribution network;

(j) if the node at top of connection address list is not part of the distribution network, deleting such node from connection address list and repeating steps (h) and (i) with respect to the next node at the top of connection address list; and

(k) if the node at top of connection address list is part of said distribution network, having said connection requesting user node dock with said node at the top of connection address list.

In another example, the process may further comprise the following steps:

(h) having said connection requesting user node go to (or attempt to go to) the node at the top of connection address list;

(i) having said connection requesting user node determine whether the node at the top of said connection address list is part of the distribution network;

(j) if the node at top of connection address list is not part of the distribution network, deleting such node from connection address list and repeating steps (h) and (i) with respect to the next node at the top of connection address list;

(k) if the node at the top of connection address list is part of said distribution network, determining whether said node at the top of connection address list has available capacity to transmit content data to said connection requesting user node; and

(l) if the node at the top of said connection address list is part of said distribution network and has available capacity to transmit content data to said connection requesting user node, having said connection requesting user node dock with the node at the top of said connection address list.

In another example, the process may further comprise the following steps:

(m) if the node at the top of said connection address list is part of said distribution network and the node at the top of said connection address list does not have available capacity to transmit content data to said connection requesting user node, repeating steps (a)-(g), with the node at the top of said connection address list being said instructing node and a new connection address list being the product of steps (a)-(g).

In another example, the process may further comprise the steps of counting the times that step (m) is performed and, when step (m) is performed a predetermined number of times, having said connection requesting user node repeat steps (a)-(g) with said primary server being the instructing node.

In another embodiment a process for docking a connection requesting user node with a distribution network is provided, said process including the following steps:

(a) having an instructing node receive a connection request from said connection requesting user node;

(b) determining whether said connection requesting user node's bandwidth capacity is below a predetermined threshold (e.g., said connection requesting user node is a low-bandwidth node) or at least said predetermined threshold (e.g., said connection requesting user node is a high-bandwidth node);

(c) forming an instructing node topology database indicating, at a point in time:

(i) which, if any, user nodes are docked with said instructing node as child nodes; which, if any, user nodes are docked with each of said child nodes as grandchild nodes of said instructing node; which, if any, user nodes are docked with each of said grandchild nodes as great-grandchild nodes of said server node; and so on; and

(ii) said docked user nodes' respective bandwidth capacities;

(d) if said connection requesting user node is a high-bandwidth user node, selecting from a group of nodes including the instructing node and said docked user nodes a recommended parent node for said connection requesting user node, wherein said recommended parent node has apparent available capacity to transmit content data to said connection requesting user node and is at least as close (i.e., in terms of network topology) to said instructing node as any other docked user node having apparent available capacity to transmit content data to said connection requesting user node;

(e) if said connection requesting user node is a low-bandwidth user node, selecting from a group of nodes including the instructing node and said docked user nodes a recommended parent node for said connection requesting user node, wherein said recommended parent node has apparent available capacity to transmit content data to said connection requesting user node and is at least as far (i.e., in terms of network topology) from said instructing node as any other docked user node having apparent available capacity to transmit content data to said connection requesting user node; and

(f) providing a connection address list to said connection requesting user node, said connection address list listing each node in said instructing node's topology database from said recommended parent node back to said instructing node.

In one example, step (f) may further comprise the following step: (i) if said instructing node is not a server node, including in said connection address list a list of nodes cascadingly connected with each other from the instructing node back to said server node.

In another embodiment a process for connecting a connection requesting user node to a computer information distribution network having a primary server node and user nodes docked therewith in a cascaded relationship is provided, said process further comprising the following steps:

(a) providing said connection requesting user node with a connection address list which sets forth a list of user nodes docked in series with each other back to said primary server;

(b) having said connection requesting user node go to (or attempt to go to) the node at top of said connection address list;

(c) having said connection requesting user node determine whether the node at the top of said connection address list is part of the distribution network;

(d) if the node at the top of said connection address list is not part of the distribution network, deleting such node from said connection address list and repeating steps (b) and (c) with respect to the next node at the top of connection address list; and

(e) if the node at the top of said connection address list is part of said distribution network, having said connection requesting user node dock with the node at the top of said connection address list.

In another embodiment a process for connecting a connection requesting user node to a computer information distribution network having a primary server node and user nodes docked therewith in a cascaded relationship is provided, said process further comprising the following steps:

(a) providing said connection requesting user node with a connection address list which sets forth a list of user nodes docked in series with each other back to said primary server;

(b) having said connection requesting user node go to (or attempt to go to) the node at the top of said connection address list;

(c) having said connection requesting user node determine whether the node at the top of said connection address list is part of the distribution network;

(d) if the node at the top of said connection address list is not part of the distribution network, deleting such node from connection address list and repeating steps (b) and (c) with respect to the next node at the top of said connection address list;

(e) if the node at the top of said connection address list is part of said distribution network, determining whether the node at the top of said connection address list has available capacity to transmit content data to said connection requesting user node; and

(f) if the node at the top of said connection address list is part of said distribution network and has available capacity to transmit content data to said connection requesting user node, having said connection requesting user node dock with the node at the top of said connection address list.

In another embodiment relating to a computer information distribution network comprising a primary server node and user nodes docked therewith in a cascaded relationship, wherein each user node may have no more than a predetermined maximum number of child nodes docked with it, a process for reconfiguring the network in the event of a user node's departure therefrom is provided, said process including the following steps:

(a) providing information at a point in time to a node in the distribution network as follows:

(i) the node's propagation rating if it is a user node having a parent node which is a user node, wherein a propagation rating is one of a predetermined number of grades ranging from highest to second highest to third highest and so on, with the number of grades being equal to said predetermined maximum number;

(ii) an ancestor list setting forth the node's ancestor nodes' addresses (if it has any ancestor nodes) back to the primary server node, with the node's parent node being atop the ancestor list;

(iii) a sibling list of the node's sibling nodes' addresses (if it has any sibling nodes) and their respective propagation ratings (of note, in another example (which example is intended to be illustrative and not restrictive), nodes may not have data about their sibling (i.e., nodes do not know who their sibling is); instead, the mechanism used to get a red node to connect to its green sibling may be for the parent of those two nodes (or a proxy for that parent) to send the red child: (1) a connection path with that red child's green sibling at the top of the path; and (2) a command to "priority join" to that green child - when the green child node accepts the incoming "priority join" connection request from its former red sibling, the green node will kick its own red child (if such a child exists) instructing that red child to join as a child of its own green sibling (these actions will cause a chain reaction of reconfigurations which will cascade until the edge of the tree is reached)); and

(iv) a child list of the node's child nodes' addresses (if it has any child nodes) and their respective propagation ratings;

(b) having a first node send a propagation signal to each former child node of a departed node, wherein said first node is an ancestor of said former child node of said departed node;

(c) with respect to each child node receiving the propagation signal which did not have the highest propagation rating prior to its receiving the propagation signal (hereinafter referred to as a "red node"), upon its receiving the propagation signal:

(i) setting the propagation rating of the red node to the next higher grade above the propagation rating of that red node before it received the propagation signal;

(ii) having the red node undock from its parent node (if it was docked with its parent node before it received the propagation signal); and

(iii) having the red node dock with its sibling node which had the highest propagation rating prior to the red node's receiving the propagation signal;

(d) with respect to each child node receiving the propagation signal which had the highest propagation rating prior to its receiving a propagation signal (hereinafter referred to as a "green node"), upon its receiving the propagation signal:

(i) setting the propagation rating of that green node to the lowest grade;

(ii) having the green node retransmit the propagation signal to its child nodes (if it has any); and

(iii) determining whether the green node is docked with the node from which the green node received the propagation signal and if it is, having the green node remain docked with said node from which the green node received the propagation signal, and if it is not, having the green node dock with said node from which the green node received the propagation signal; and

(e) repeating steps (c) and (d) with respect to each user node which receives a retransmitted propagation signal.

In another embodiment relating to a computer information distribution network comprising a primary server node and user nodes docked therewith in a cascaded relationship, wherein each user node may have no more than a predetermined maximum number of child nodes docked with it, a process for reconfiguring the network in the event of a user node's departure therefrom is provided, said process including the following steps:

(a) providing information at a point in time to a node in the distribution network as follows:

(i) the node's propagation rating if it is a user node having a parent node which is a user node, wherein a propagation rating is one of a predetermined number of grades ranging from highest to second highest to third highest and so on, with the number of grades being equal to said predetermined maximum number;

(ii) an ancestor list setting forth the node's ancestor nodes' addresses (if it has any ancestor nodes) back to the primary server node, with the node's parent node being atop the ancestor list;

(iii) a sibling list of the node's sibling nodes' addresses (if it has any sibling nodes) and their respective propagation ratings; and

(iv) a child list of the node's child nodes' addresses (if it has any child nodes) and their respective propagation ratings;

(b) having a first node send a propagation signal to each former child node of a departed node, wherein said first node is an ancestor of said former child node of said departed node;

(c) with respect to each child node receiving the propagation signal which did not have the highest propagation rating prior to its receiving the propagation signal (hereinafter referred to as a "red node"), upon its receiving the propagation signal:

(i) setting the propagation rating of the red node to the next higher grade above the propagation rating of that red node before it received the propagation signal;

(ii) having the red node undock from its parent node (if it was docked with its parent node before it received the propagation signal);

(iii) devising a first connection address list comprising the address of the red node's sibling node which had the had the highest propagation rating prior to the red node's receiving the propagation signal followed by the red node's ancestor list;

(iv) having said red node go to (or attempt to go to) the node at the top of said first connection address list;

(v) having said red node determine whether the node at the top of said first connection address list is part of the distribution network;

(vi) if the node at the top of said first connection address list is not part of the distribution network, deleting such node from said first connection address list and repeating steps (c)(iv) and (c)(v) with respect to the next node at the top of said first connection address list;

(vii) if the node at the top of said first connection address list is part of said distribution network, determining whether the node at the top of said first connection address list has available capacity to transmit content data to said red node; and

(viii) if the node at the top of said first connection address list is part of said distribution network and has available capacity to transmit content data to said red node, having said red node dock with the node at the top of said first connection address list (with regard to these steps (vii) and (viii), in another example (which example is intended to be illustrative and not restrictive), when a red node connects to a green node as part of a network reconfiguration event, it does not matter whether there is "available capacity" or not for the green node to accept its incoming red sibling – the green node has no choice but to accept the node (these actions are facilitated in this example by the red node issuing a "priority join" to the green node; if the green node does not have "available capacity" it has to kick its own red child to make room for the incoming green node));

(d) with respect to each child node receiving the propagation signal which had the highest propagation rating prior to its receiving a propagation signal (hereinafter referred to as a "green node"), upon its receiving the propagation signal:

(i) setting the propagation rating of that green node to the lowest grade;

(ii) having the green node retransmit the propagation signal to its child nodes (if it has any);

(iii) determining whether the green node is docked with the node from which the green node received the propagation signal and if it is, having the green node remain docked with said node from which the green node received the propagation signal, and if it is not, devising a second connection address list starting with said node from which the green node received the propagation signal and followed by that portion of said green node's ancestor list extending back to the primary server from said node from which the green node received the propagation signal;

(iv) having said green node go to (or attempt to go to) the node at the top of said second connection address list;

(v) having said green node determine whether the node at the top of said second connection address list is part of the distribution network;

(vi) if the node at the top of said second connection address list is not part of the distribution network, deleting such node from said second connection address list and repeating steps (d)(iv) and (d)(v) with respect to the next node at the top of said second connection address list;

(vii) if the node at the top of said second connection address list is part of said distribution network, determining whether said node at the top of said second connection address list has available capacity to transmit content data to said green node; and

(viii) if the node at the top of said second connection address list is part of said distribution network and has available capacity to transmit content data to said green node, having said green node dock with the node at the top of said second connection address list; and

(e) repeating steps (c) and (d) with respect to each user node which receives a retransmitted propagation signal.

In one example, step (c)(iii) may include the following additional step: (1) waiting a predetermined period of time so as to allow the red node's sibling with the highest

propagation rating prior to the red node's receipt of the propagation signal (hereinafter the "green node") to retransmit the propagation signal to the green node's child nodes (if any).

In another embodiment relating to a computer information distribution network comprising an instructing node having a fist user node docked with said instructing node to receive content data therefrom and a second user node docked with said first user node to receive content data therefrom, a process for handling a purported communication interruption between said first user node and said second user node is provided, said process including the following steps:

(a) having said instructing node receive a first complaint about said first user node from said second user node;

(b) if the second user node is the only user node docked with said first user node at a point in time, if said instructing node receives another complaint about the first user node from another user node within a predetermined period of time from when said instructing node received said first complaint or if said instructing node experiences a communication problem between it and said first user node during said predetermined period of time: (i) disconnecting said first user node from said instructing node; and

(c) if said instructing node has not experienced a communication problem between it and said first user node during said predetermined period of time and if a plurality of user nodes are docked with said first user node at said point in time and said instructing node has not received another complaint about said first user node from a user node other than said second user node within said predetermined period of time: (i) disconnecting said second user node from said first user node.

In one example, step (b) may include the following step: (ii) sending a signal to the user nodes docked with said first user node indicating the departure of said first user node from said network.

In another example, each user node may have no more than a predetermined maximum number of child nodes docked with it, and wherein step (b) may include the following steps

(ii) providing information at said point in time to each of said second user node, the other user nodes which had been docked with said first user node (if any) and at least one of any user nodes cascadingly connected in the distribution network under said first user node prior to said first user node's being disconnected from said instructing node (if any) as follows:

(1) the node's propagation rating, wherein a propagation rating is one of a predetermined number of grades ranging from highest to second highest to third highest and so on, with the number of grades being equal to said predetermined maximum number;

(2) an ancestor list setting forth the node's ancestor nodes' addresses (if it has any ancestor nodes) back to the instructing node, with the node's parent node being atop the ancestor list;

(3) a sibling list of the node's sibling nodes' addresses (if it has any sibling nodes) and their respective propagation ratings; and

(4) a child list of the node's child nodes' addresses (if it has any child nodes) and their respective propagation ratings;

(iii) having said instructing node send a propagation signal to each of said second user node and the other user nodes which had been docked with said first user node (if any);

(iv) with respect to each user node receiving the propagation signal which did not have the highest propagation rating prior to its receiving the propagation signal (hereinafter referred to as a "red node"), upon its receiving the propagation signal:

(1) setting the propagation rating of the red node to the next higher grade above the propagation rating of that red node before it received the propagation signal;

(2) having the red node undock from its parent node (if it was docked with its parent node before it received the propagation signal); and

(3) having the red node dock with its sibling node which had the highest propagation rating prior to the red node's receiving the propagation signal;

(v) with respect to each user node receiving the propagation signal which had the highest propagation rating prior to its receiving a propagation signal (hereinafter referred to as a "green node"), upon its receiving the propagation signal:

(1) setting the propagation rating of that green node to the lowest grade;

(2) having the green node retransmit the propagation signal to its child nodes (if it has any); and

(3) determining whether the green node is docked with the node from which the green node received the propagation signal and if it is, having the green node remain docked with said node from which the green node received the propagation signal, and if it is not, having the green node dock with said node from which the green node received the propagation signal; and

(vi) repeating steps (iv) and (v) with respect to each user node which receives a retransmitted propagation signal.

In another embodiment a distribution network for the distribution of content data from a server node to user nodes, wherein said user nodes are connected to said server and each other in cascaded relationship is provided, wherein:

(a) at least one of said user nodes is a repeater node connected directly to said server node, wherein said repeater node retransmits content data received by it to a user node docked to it for purpose of receiving content data from said repeater node (hereinafter referred to as a "child node"); and

(b) wherein each repeater node has the ability to provide to a user node which is attempting to dock with said repeater node connection address instructions.

In one example, each repeater node may include a descendant database indicating:

(i) which child nodes, if any, at a point in time, are docked with it so as to receive content data from said repeater node, and

(ii) which user nodes, if any, at said point in time, are purportedly docked with each of said child nodes.

In another example, the connection address instructions may refer said user node which is attempting to dock with said repeater node to a node in said descendant database.

In another example, each user node may include an ancestor database indicating to which node said user node is docked so that said user node may receive content data therefrom (hereinafter referred to as a "parent node"), and to which node, if any, at said point in time, said parent node is docked so that it may receive content data therefrom.

In another example, if said parent node of said user node departs from said distribution network, said user node may contact another node on its ancestor database.

In another example, each child node of a repeater node may include a sibling database indicating which user nodes, if any, are also child nodes of said repeater node.

In another embodiment a distribution network for the distribution of content data from a server node to user nodes is provided, wherein n levels of user nodes are cascadingly connected to said server node, wherein n is a number greater than one, wherein each user node includes a delay which causes the playing of content data by such node to be delayed a period of time (hereinafter "delay time") from a point in time, and wherein delays in higher level nodes create greater delay times than do delays in lower level nodes.

In another embodiment a distribution network for the distribution of content data from a server node to user nodes is provided, wherein n levels of user nodes are cascadingly

connected to said server node, wherein n is a number greater than one, wherein each user node includes a delay which causes the playing of content data by such node to be delayed a period of time (hereinafter "delay time") from a point in time, wherein x is a number from and including 2 to and including n, and wherein delays in (x-1) level nodes create greater delay times than do delays in x level nodes.

In one example, said point in time is an event experienced approximately simultaneously by substantially all user nodes.

In another example, said point in time is when said node receives content data.

While a number of embodiments and examples of the present invention have been described, it is understood that these embodiments and examples are illustrative only, and not restrictive, and that many modifications may become apparent to those of ordinary skill in the art. For example, any desired branch factor may be used (*see*, e.g., FIG. 46A showing a branch factor of 1 (i.e., each parent node may have only one child node); FIG. 46B showing a branch factor of 2 (i.e., each parent node may have one or two children nodes); and FIG. 46C showing a branch factor of 3 (i.e., each parent node may have one, two or three children nodes)). Of course, other branch factors (e.g., 4, 5, 6, etc) may be used and, as discussed above, the distribution network may be a hybrid network utilizing subtrees having different branch factors). Further still, a modified Utility Rating may be determined as follows: Utility Rating = Time (of connection) x Bandwidth Rating. Further still, a node ID may be based upon: (a) an IP address (external/internal/DHCP); (b) a URL; and/or (c) a Port. Further still, the root server may propagate "its" time upon assignment of a connection and the user node may get a "delta" (i.e., difference) from its clock to use (in the event of a reset the user node may re-acquire the time from the server (or from another time-providing source)). Further still, the various time intervals discussed herein may be predetermined time intervals (e.g., .5 sec., 1 sec, 1 min.) and/or the time intervals may be based upon some varying criteria (e.g., flow of data in the network, number of nodes in the network, number of levels in the network, position of a given node in the network). Further still, partial propagation of data (e.g., ancestor data and/or descendent data) from one or more nodes may be permitted. Further still (as discussed above), one or more of the steps described herein may be omitted (and the various steps which are performed may not necessarily need to be carried out in the order described herein (which description was intended to represent a number of examples, and not be restrictive)). Further still, the root server and each of the user nodes may utilize essentially identical software for carrying out the network configuration, reconfiguration and data transfer operations described herein or the root server may have dedicated server software

while the user nodes have different, dedicated user node software. Further still, the entire distribution network (e.g., the root server and all of the user nodes) may have the same branch factor (e.g., a branch factor of 2) or the user nodes may have a first branch factor (e.g., a branch factor of 2) and the root server may have a second branch factor (e.g., the root server may be capable of acting as a parent to more user nodes than 2). Further still, heuristics may be employed to drive network configuration, reconfiguration and/or data flow (e.g., based upon historic network configuration, reconfiguration and/or data flow).

Aylo Exhibit 1002
Page 174 of 1405

CLAIMS

1. A method for propagating a network topology model of a binary tree topology network through the network, the network including a plurality of nodes, including a root node and at least one leaf node, operatively networked together to provide a communications path for data, each of the plurality of nodes except the root node being docked downtree of a parent node, the parent node being one of the plurality of nodes, and each of the plurality of nodes adapted to be docked uptree of a first child node and a second child node, each of the at least one leaf nodes having none of the plurality of nodes docked downtree of and therewith, the method comprising the steps of:

each of the at least one leaf nodes providing a propagate network topology model message containing the address of the providing leaf node to the parent node docked uptree of and with the providing leaf node;

for each of the plurality of nodes receiving the propagate network topology model message from each child node docked downtree of and therewith:

forming a network topology model representing one or more paths from the receiving node to each one of the plurality of nodes downtree of the receiving node;

storing in a memory accessible to the receiving node, the network topology model so formed;

providing to the parent node docked uptree of and with the receiving node a propagate network topology model message containing the network topology model so formed,

thereby propagating a network topology model through the network.

2. The method of claim 1, wherein the communications path for data comprises a downtree communications path.

3. The method of claim 1, wherein the communications path for data comprises a downtree communications path for data and an uptree communications path for data.

4. The method of claim 1, wherein the step of each of the at least one leaf nodes providing a propagate network topology model message is carried out a plurality of times.

5. The method of claim 1, wherein the step of each of the at least one leaf nodes providing a propagate network topology model message is carried out periodically.

6. The method of claim 1, wherein a first node being one of the plurality of nodes, but not a leaf node, receives a register message from a second node that is not part of the network, and wherein the first node is docked uptree of a first child node and a second child node, each of the first child node and the second child node being one of the plurality of nodes, the method further comprising the steps of:

searching the network topology model stored in the memory of the first node for a path to an open node, the open node being one of the plurality of nodes that the network topology model shows as not being docked uptree of and with two child nodes;

sending a path message to the second node containing at least an address of the open node.

7. The method of claim 6, wherein the path message comprises a path from the open node to the first node.

8. The method of claim 6, wherein the searching step comprising finding the open node such that a connection to the open node keeps the binary topology network balanced.

9. The method of claim 6, wherein step of searching further comprises the steps of:

searching consecutively further down hierarchical levels of the binary tree represented in the network topology model stored in the memory of the first node:

first to locate a path representing a node not being docked uptree of and with any child nodes, and if no such path is located,

then to locate a path representing a node not being docked uptree of and with two child nodes, and

repeating the searching step until one such path is located; and

designating the located path as a path to the open node.

10. A method for propagating a network topology model of a binary tree topology network through the network, the network including a plurality of nodes, including a root node and at least one leaf node, operatively networked together to provide a communications path for data, each of the plurality of nodes except the root node being docked downtree of a parent node, the parent node being one of the plurality of nodes, and each of the plurality of nodes adapted to be docked uptree of a first child node and a second child node, each of the at least one leaf nodes having none of the plurality of nodes docked downtree of and therewith, the method comprising the steps of:

providing a propagate path message from the root node to each of the plurality of nodes that are docked downtree of and with the root node, the propagate path message indicating an address of the root node;

for each of the plurality of nodes receiving the propagate path message from the root node or from another one of the plurality of nodes:

storing in a memory accessible to the receiving node, the path to the root node stored in the propagate path message;

if the receiving node has at least one of the plurality of nodes docked downtree of and therewith, adding to the propagate path message an address of the receiving node; and

providing the propagate path message to each of the plurality of nodes docked downtree of and with the receiving node, the propagate path message a path to the root node;

thereby propagating a root path of a binary tree topology network through the network.

11. The method of claim 10, wherein the communications path for data comprises a downtree communications path for data.

12. The method of claim 10, wherein the communications path for data comprises a downtree communications path for data and an uptree communications path for data.

13. The method of claim 10, wherein the step of providing a propagate path message is carried out a plurality of times.

14. The method of claim 10, wherein the step of providing a propagate path message is carried out periodically.

15. A method for propagating a network topology model of a binary tree topology network through the network, the network including a plurality of nodes, including a root node and at least one leaf node, operatively networked together to provide a communications path for data, each of the plurality of nodes except the root node being docked downtree of a parent node, the parent node being one of the plurality of nodes, and each of the plurality of nodes adapted to be docked uptree of a first child node and a second child node, each of the at least one leaf nodes having none of the plurality of nodes docked downtree of and therewith, the method comprising the steps of:

each of the at least one leaf nodes, providing a propagate network topology model message containing the address of the providing leaf node to the parent node docked uptree of and with the providing leaf node;

for each of the plurality of nodes receiving the propagate network topology model message from each child node docked downtree of and therewith:

forming a network topology model comprising a path from the receiving node to each one of the plurality of nodes downtree of the receiving node;

storing in a memory accessible to the receiving node, the network topology model so formed;

providing to the parent node docked uptree of and with the receiving node a propagate network topology model message containing the network topology model so formed,

providing a propagate path message from the root node to each of the plurality of nodes that are docked downtree of and with the root node, the propagate path message indicating an address of the root node;

for each of the plurality of nodes receiving propagate path message from the root node or from another one of the plurality of nodes:

storing in a memory accessible to the receiving node, the path to the root node stored in the propagate path message;

if the receiving node has at least one of the plurality of nodes docked downtree of and therewith, adding to the propagate path message an address of the receiving node; and

providing the propagate path message to each of the plurality of nodes docked downtree of and with the root node.

16. The method of claim 15, wherein the communications path for data comprises a downtree communications path for data.

17. The method of claim 15, wherein the communications path for data comprises a downtree communications path for data and an uptree communications path for data.

18. The method of claim 15, wherein the step of providing a propagate network topology model message is carried out a plurality of times.

19. The method of claim 15, wherein the step of providing a propagate network topology model message is carried out periodically.

20. The method of claim 15, wherein the step of providing a propagate path message is carried out a plurality of times.

21. The method of claim 15, wherein the step of providing a propagate path message is carried out periodically.

22. The method of claim 15, wherein a first node being one of the plurality of nodes, but not a leaf node, receives a register message from a second node that is not part of the network, and wherein the first node is docked uptree of a first child node and a second child node, each of the first child node and the second child node being one of the plurality of nodes, the method further comprising the steps of:

searching the network topology model stored in the memory of the first node for a path to an open node, the open node being one of the plurality of nodes that the network topology model shows as not being docked uptree of and with two child nodes;

sending a path message to the second node containing at least an address of the open node.

23. The method of claim 22, wherein the path message comprises a path from the open node to the first node.

24. The method of claim 22, wherein the path message comprises a path from the open node to the root node.

25. The method of claim 22, wherein the searching step comprising finding the open node such that a connection to the open node keeps the binary topology network balanced.

26. The method of claim 22, wherein step of searching further comprises the steps of:

searching consecutively further down hierarchical levels of the binary tree represented in the network topology model stored in the memory of the first node:

first to locate a path representing a node not being docked uptree of and with any child nodes, and if no such path is located,

then to locate a path representing a node not being docked uptree of and with two child nodes, and

repeating the searching step until one such path is located; and

designating the located path as a path to the open node.

27. A method for propagating a network topology model of a binary tree topology network through the network, the network including a plurality of nodes, including a root node and at least one leaf node, operatively networked together to provide a communications path for data, each of the plurality of nodes except the root node being docked downtree of a parent node, the parent node being one of the plurality of nodes, and each of the plurality of nodes adapted to be docked uptree of a first child node and a second child node, each of the at least one leaf nodes having none of the plurality of nodes docked downtree of and therewith, the method comprising the steps of:

providing a banned list message from the root node to each of the plurality of nodes that are docked downtree of and with the root node, the banned list message comprising an identifier for each of a plurality of banned nodes;

for each of the plurality of nodes receiving the banned list message from the root node or from another one of the plurality of nodes:

storing in a memory accessible to the receiving node, the identifiers for each of the plurality of banned nodes; and

providing the banned list message to each of the plurality of nodes docked downtree of and with the receiving node;

thereby propagating the list of banned nodes through the network.

28. The method of claim 27, wherein the communications path for data comprises a downtree communications path for data.

29. The method of claim 27, wherein the communications path for data comprises a downtree communications path for data and an uptree communications path for data.

30. The method of claim27, further comprising the step of rejecting a register message from a new node that is not part of the network if the node is identified on the banned list.

31. A method for propagating a network topology model of a binary tree topology network through the network, the network including a plurality of nodes, including a root node and at least one leaf node, operatively networked together to provide a communications path for data, each of the plurality of nodes except the root node being docked downtree of a parent node, the parent node being one of the plurality of nodes, and each of the plurality of nodes adapted to be docked uptree of a first child node and a second child node, each of the at least one leaf nodes having none of the plurality of nodes docked downtree of and therewith, the method comprising the steps of:

the first node sending a request firewall status message over a first channel to one of the plurality of nodes that is docked with the first node;

the one of the plurality of nodes receiving the request firewall status message sending at least one response message to the first node over a separate channel; and

the first node determining its firewall status based upon whether it receives the response message.

32. The method of claim 31, wherein the communications path for data comprises a downtree communications path for data.

33. The method of claim 31, wherein the communications path for data comprises a downtree communications path for data and an uptree communications path for data.

34. The method of claim 31, wherein the step of sending at least one response message comprises the steps of:

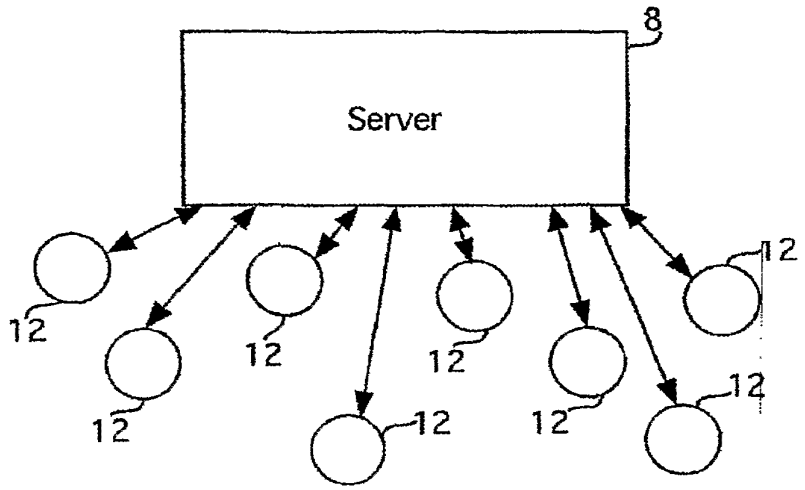
sending a first response message to the first node over a separate channel;

waiting a predefined period for an acknowledgement of the first response message, and if no acknowledgement is received, sending a second response message to the first node over the separate channel;

waiting a predefined period for an acknowledgement of the second response message, and if no acknowledgement is received, sending a third response message to the first node over the separate channel; and

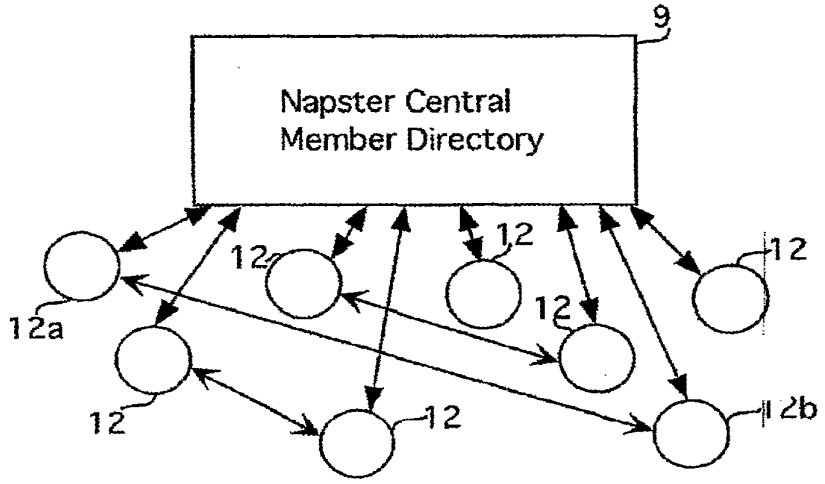
waiting a predefined period for an acknowledgement of the third response message, and if no acknowledgement is received, sending a firewall status on message to the first node over the first channel.

Aylo Exhibit 1002
Page 183 of 1405



PRIOR ART

Fig. 1



PRIOR ART

Fig. 2

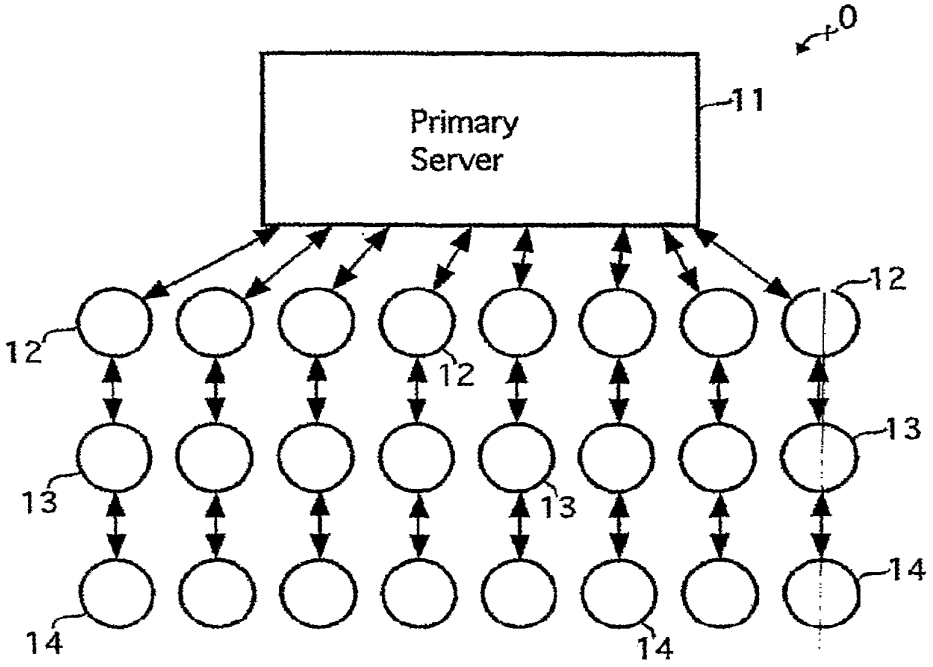


Fig. 3

"Start" [illegible]

[illegible]

Corresponds to [illegible] application [illegible]

31

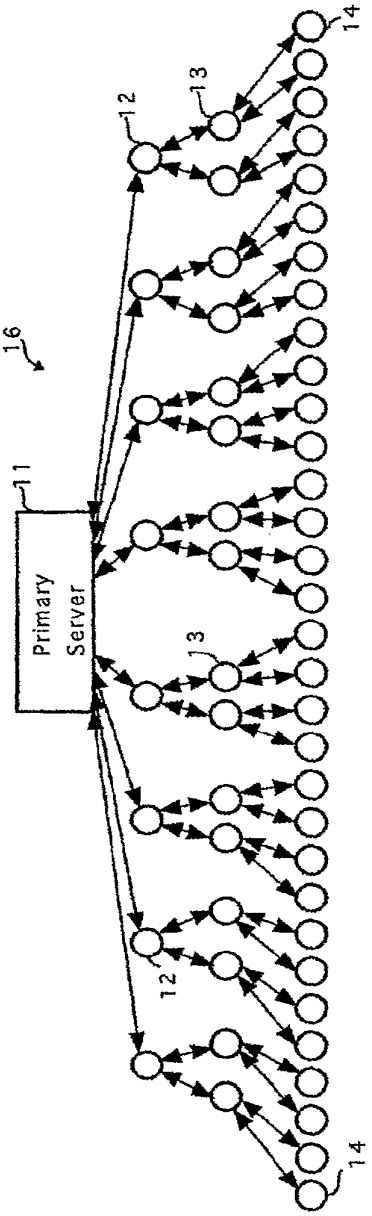


Fig. 4

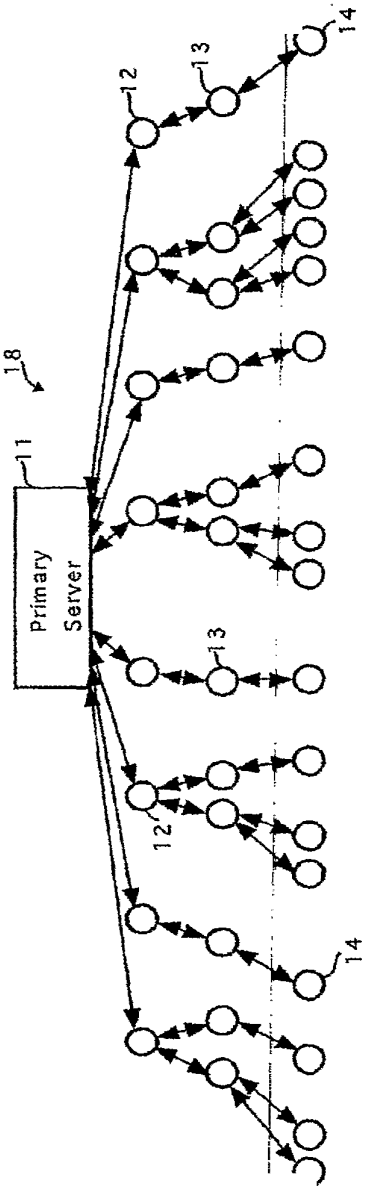


Fig. 5

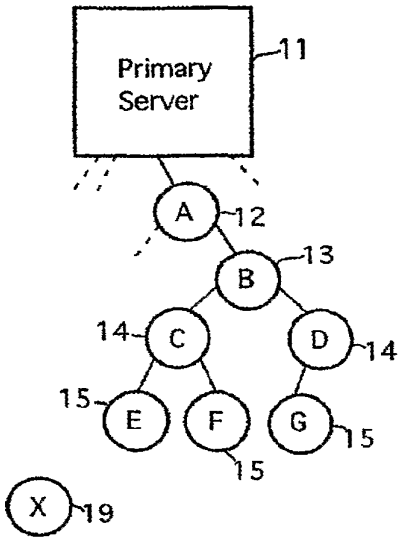


Fig. 6

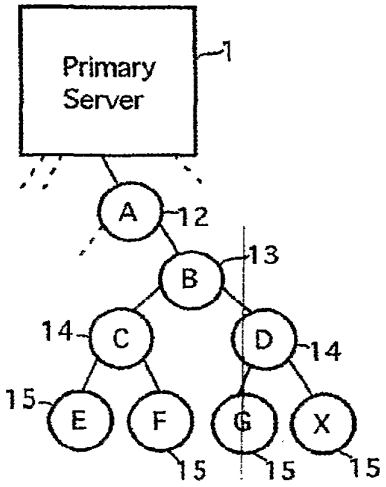


Fig. 7

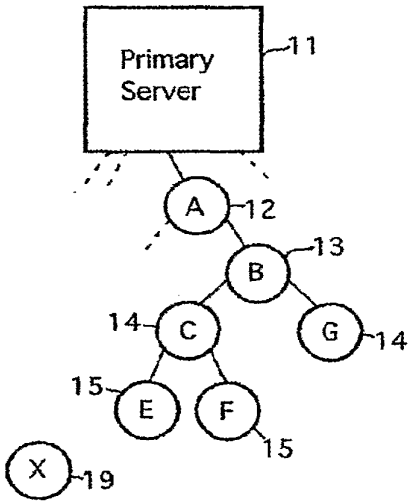


Fig. 8

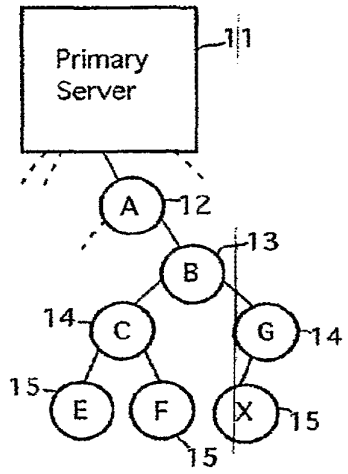


Fig. 9

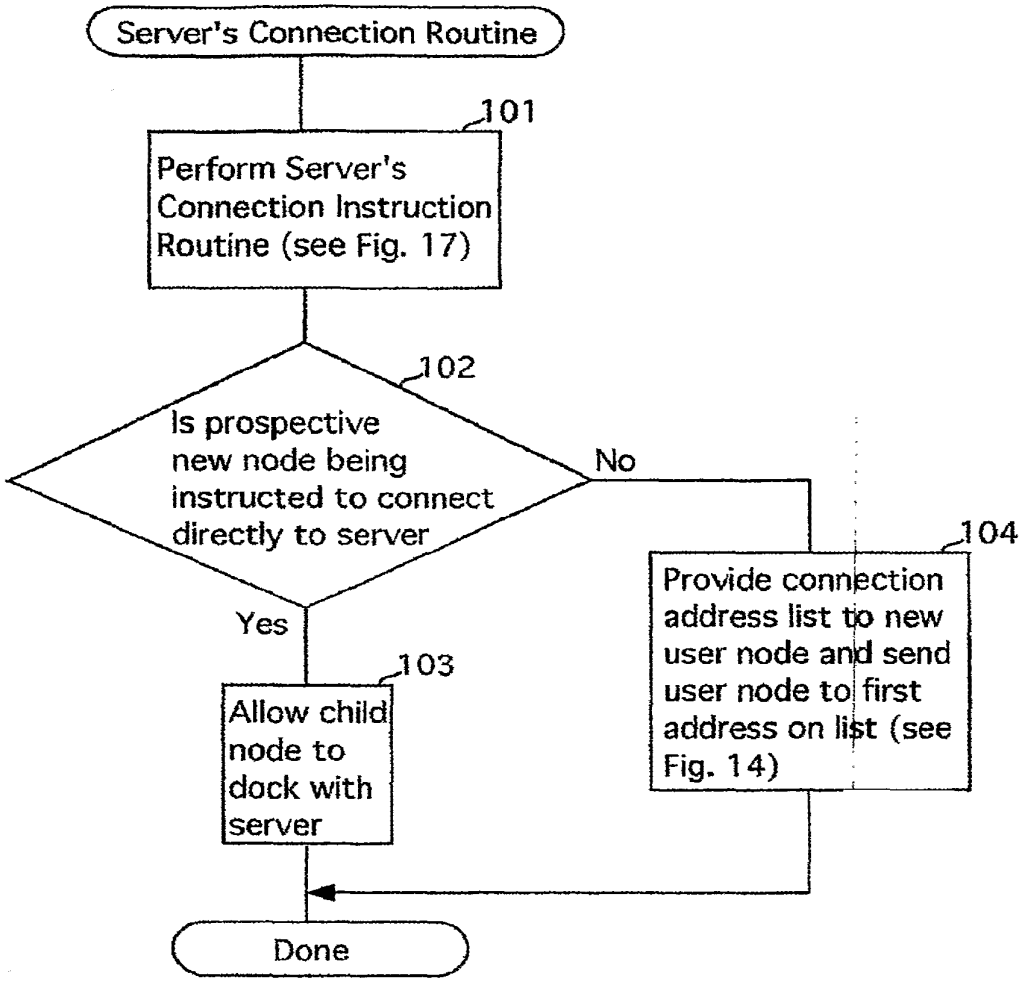

Server's Connection Routine
101
Perform Server's Connection Instruction Routine (see Fig. 17)
102
Is prospective new node being instructed to connect directly to server
No
Yes
103
Allow child node to dock with server
104
Provide connection address list to new user node and send user node to first address on list (see Fig. 14)
Done

Fig. 10

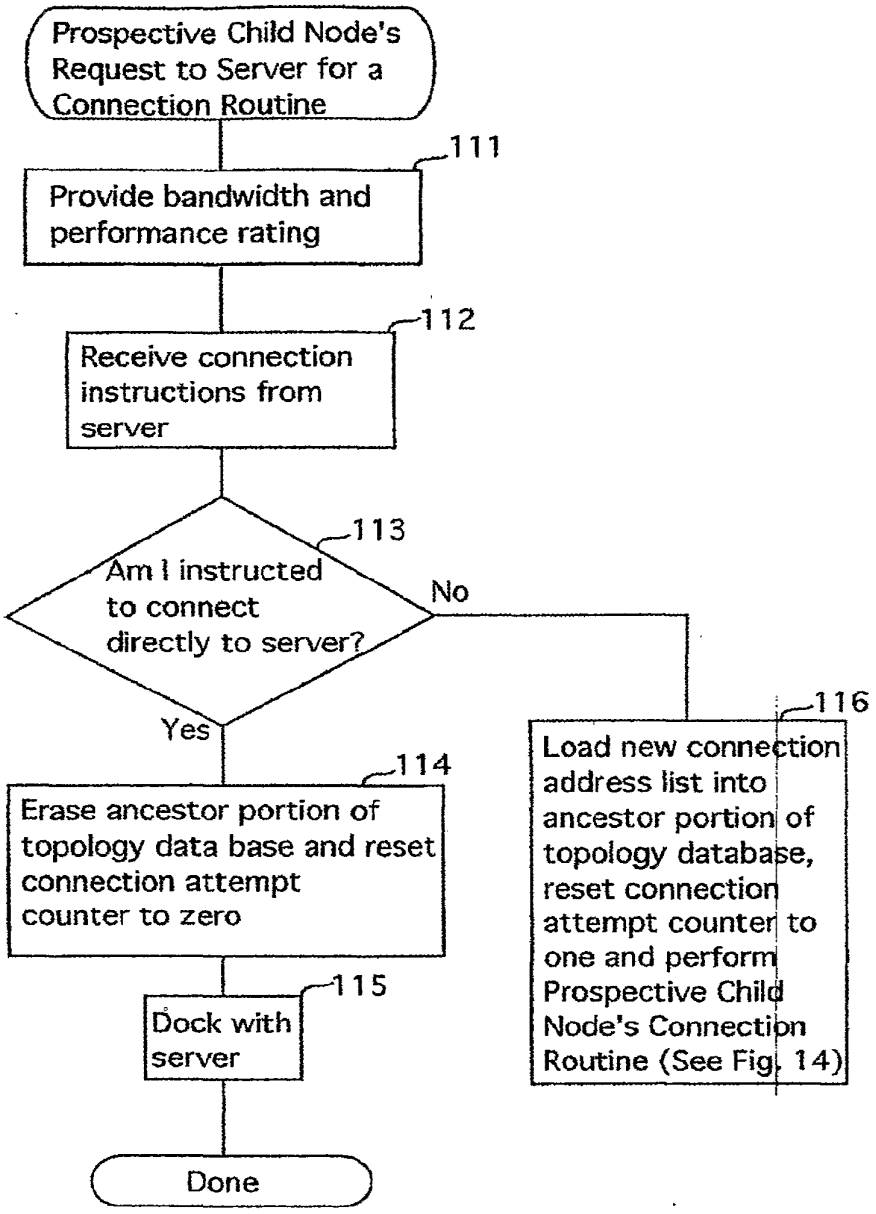

Prospective Child Node's Request to Server for a Connection Routine
111
Provide bandwidth and performance rating
112
Receive connection instructions from server
113
Am I instructed to connect directly to server?
No
Yes
114
Erase ancestor portion of topology data base and reset connection attempt counter to zero
115
Dock with server
Done
116
Load new connection address list into ancestor portion of topology database, reset connection attempt counter to one and perform Prospective Child Node's Connection Routine (See Fig. 14)

Fig. 11

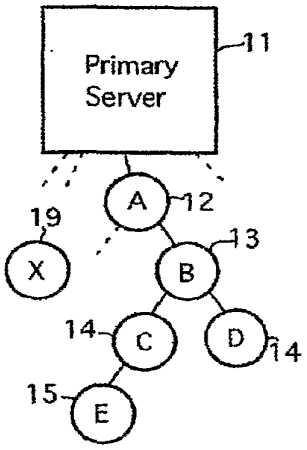


Fig. 12A

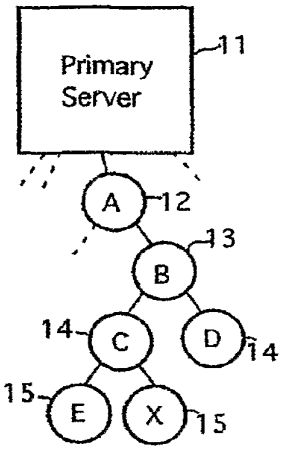


Fig. 12B

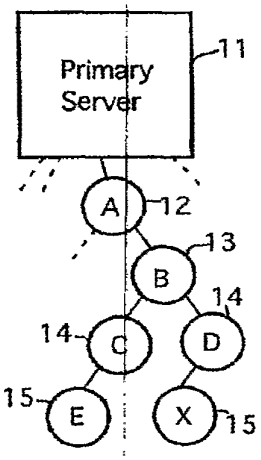


Fig. 12C

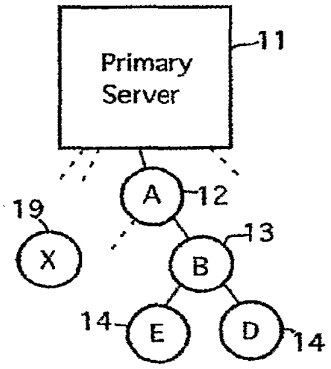


Fig. 12D

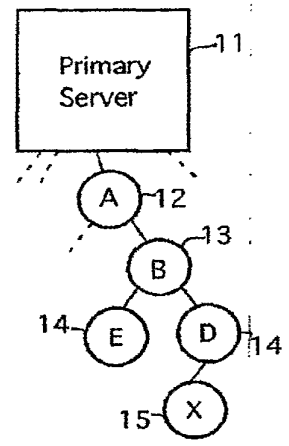


Fig. 12E

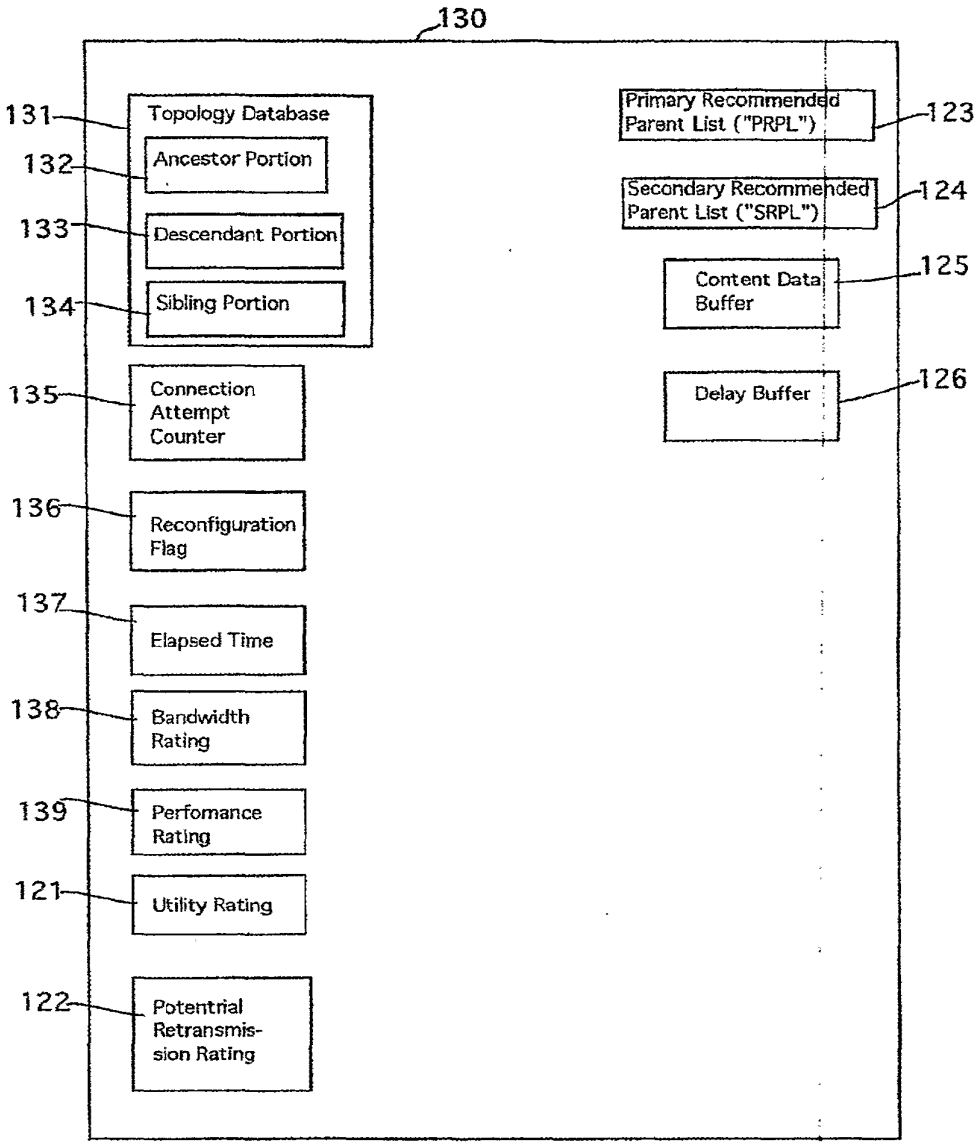

130
131
Topology Database
132
Ancestor Portion
133
Descendant Portion
134
Sibling Portion
135
Connection Attempt Counter
136
Reconfiguration Flag
137
Elapsed Time
138
Bandwidth Rating
139
Perfomance Rating
121
Utility Rating
122
Potentrial Retransmis-sion Rating
Primary Recommended Parent List ("PRPL")
123
Secondary Recommended Parent List ("SRPL")
124
Content Data Buffer
125
Delay Buffer
126

Fig. 13

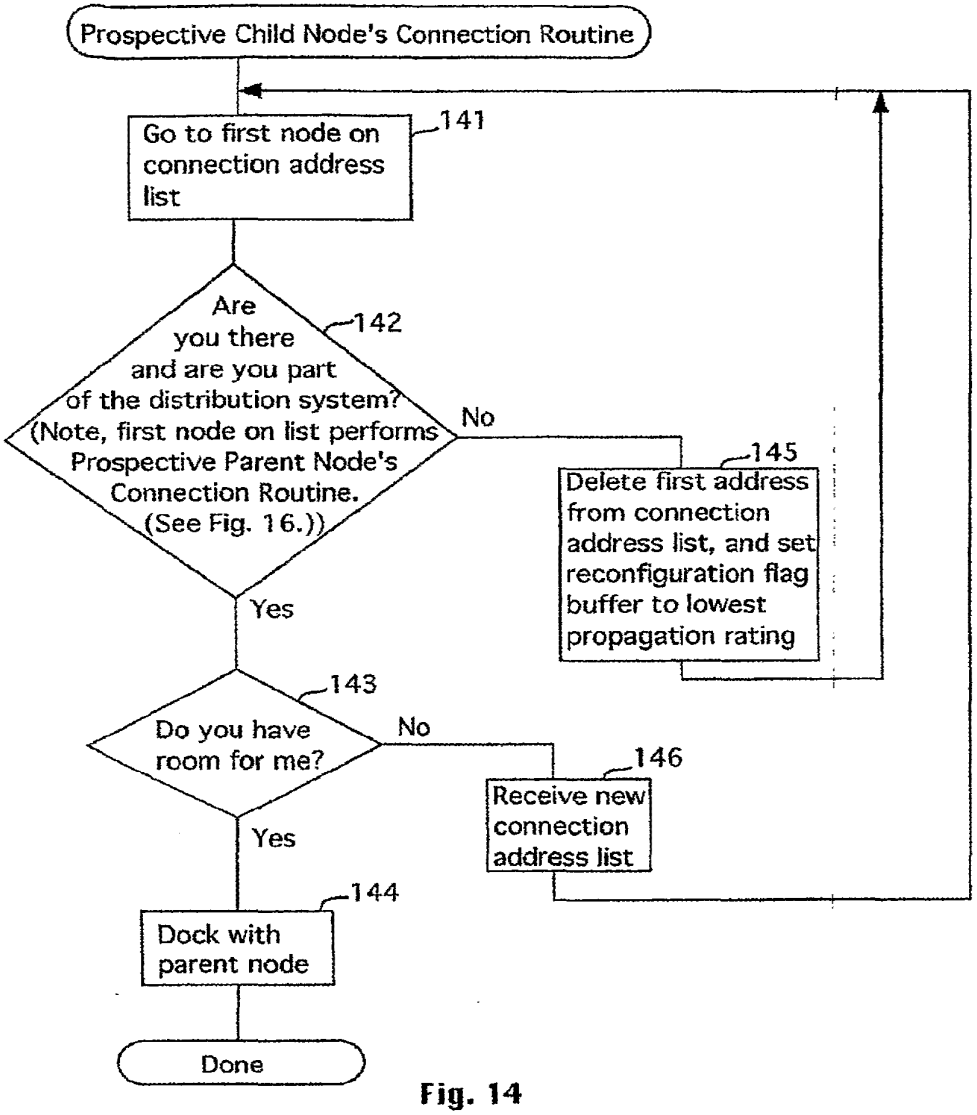

Prospective Child Node's Connection Routine
Go to first node on connection address list
141
Are you there and are you part of the distribution system? (Note, first node on list performs Prospective Parent Node's Connection Routine. (See Fig. 16.))
142
No
145
Delete first address from connection address list, and set reconfiguration flag buffer to lowest propagation rating
Yes
Do you have room for me?
143
No
146
Receive new connection address list
Yes
Dock with parent node
144
Done

Fig. 14

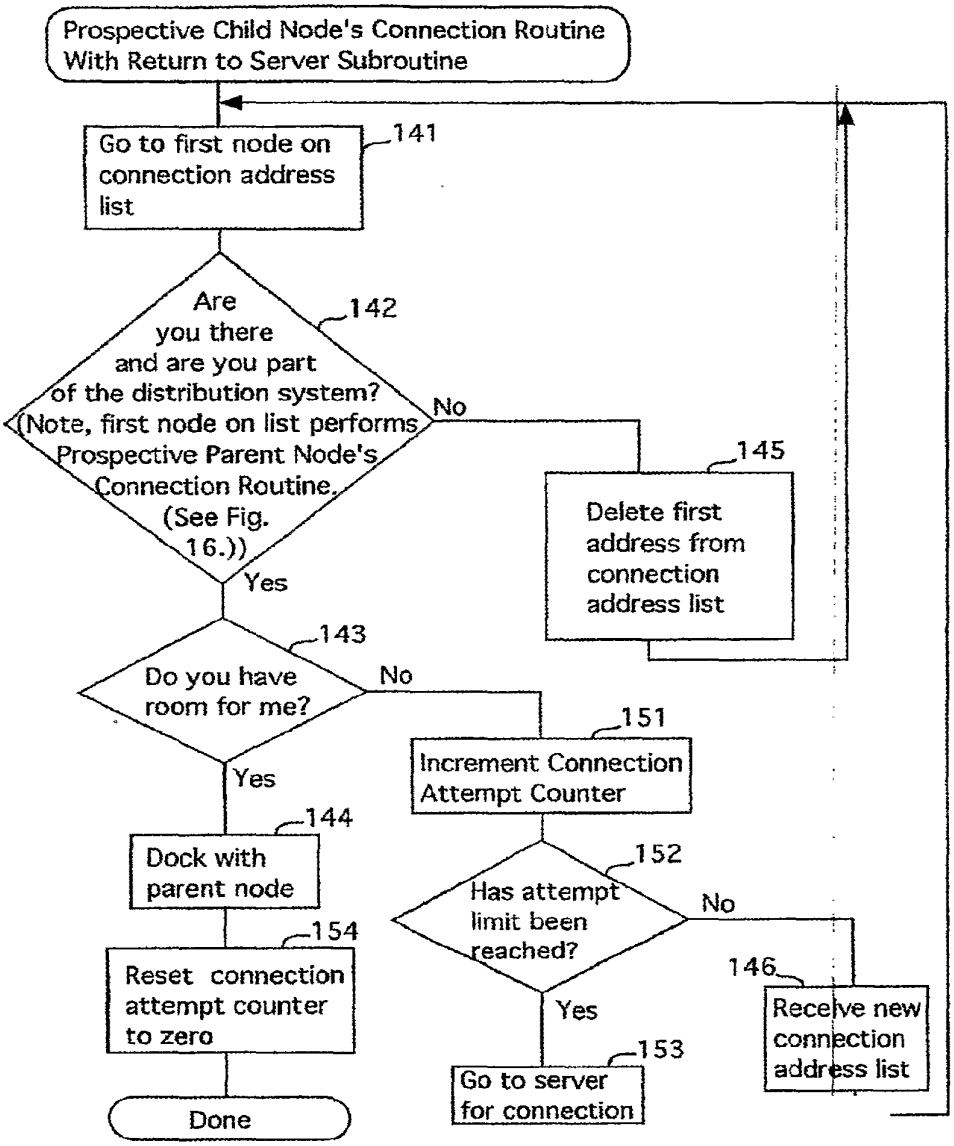

Prospective Child Node's Connection Routine With Return to Server Subroutine
Go to first node on connection address list
141
Are you there and are you part of the distribution system? (Note, first node on list performs Prospective Parent Node's Connection Routine. (See Fig. 16.))
142
No
Yes
Delete first address from connection address list
145
Do you have room for me?
143
No
Yes
Increment Connection Attempt Counter
151
Dock with parent node
144
Has attempt limit been reached?
152
No
Yes
Reset connection attempt counter to zero
154
Receive new connection address list
146
Go to server for connection
153
Done

Fig. 15

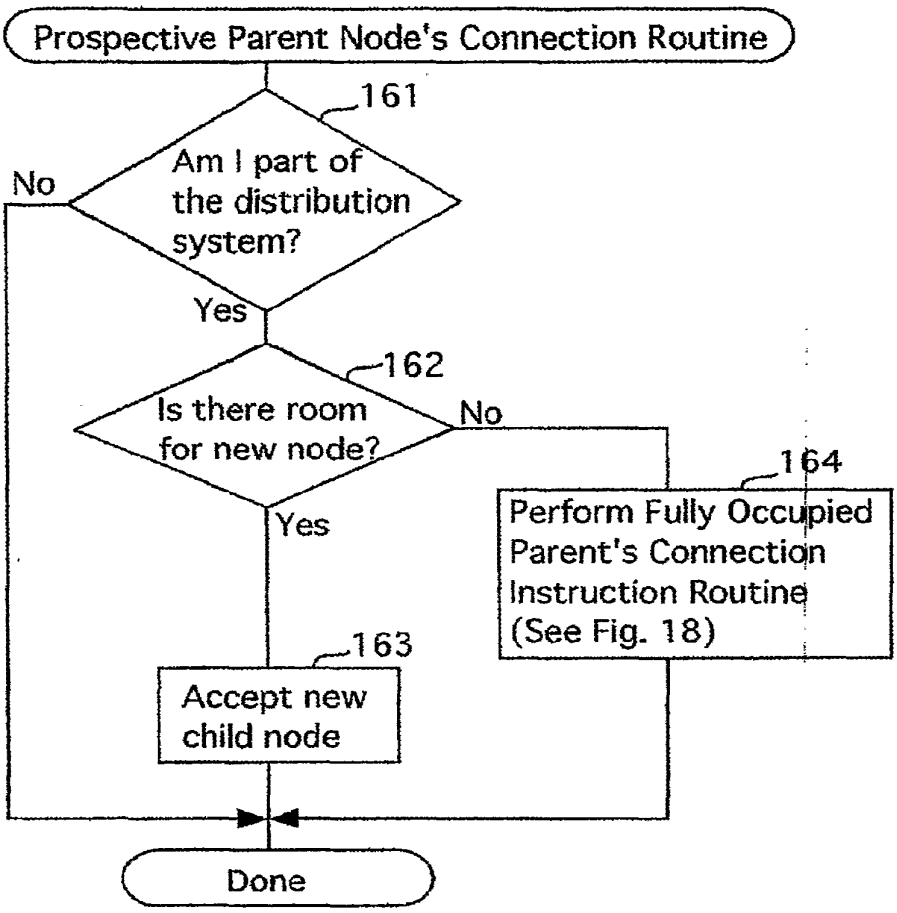

Prospective Parent Node's Connection Routine
161
Am I part of the distribution system?
No
Yes
162
Is there room for new node?
No
Yes
164
Perform Fully Occupied Parent's Connection Instruction Routine (See Fig. 18)
163
Accept new child node
Done

Fig. 16

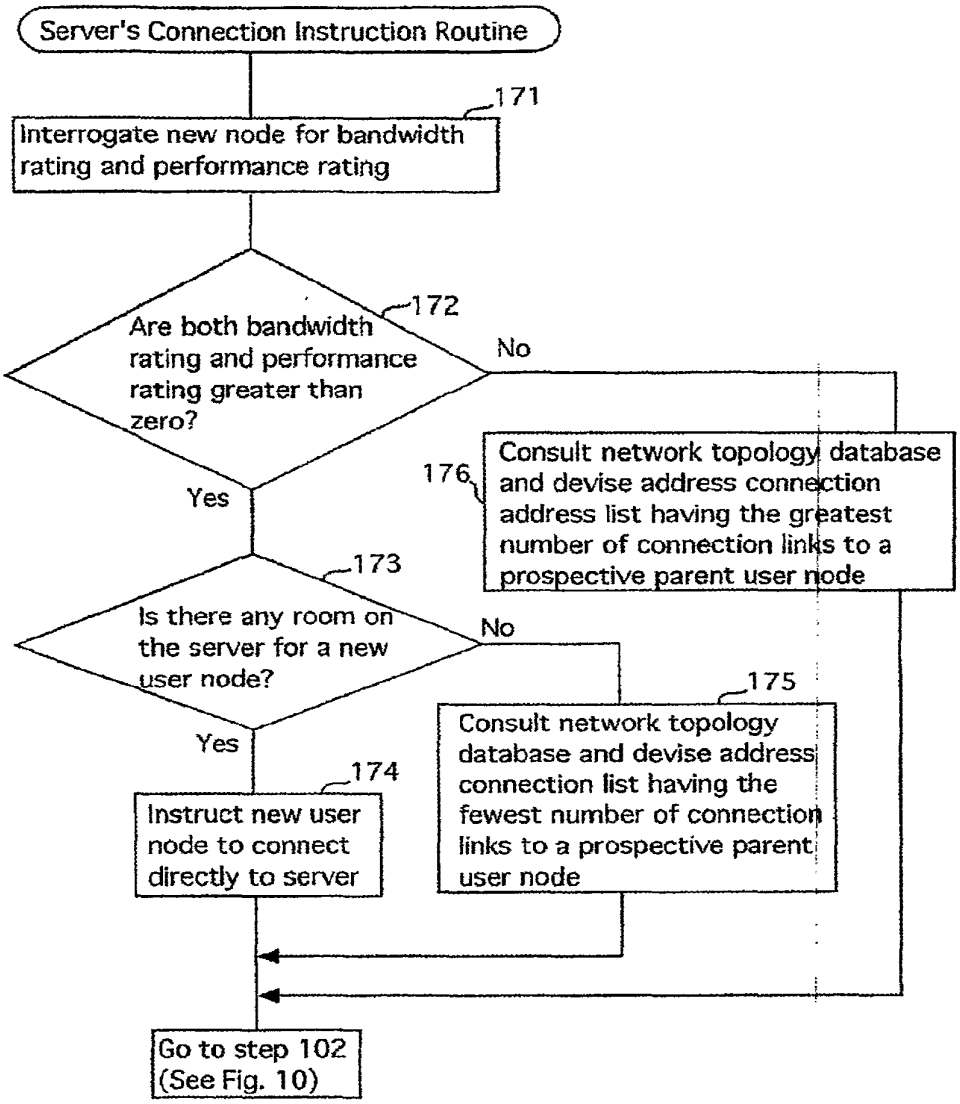

Server's Connection Instruction Routine
171
Interrogate new node for bandwidth rating and performance rating
172
Are both bandwidth rating and performance rating greater than zero?
No
Yes
176
Consult network topology database and devise address connection address list having the greatest number of connection links to a prospective parent user node
173
Is there any room on the server for a new user node?
No
Yes
175
Consult network topology database and devise address connection list having the fewest number of connection links to a prospective parent user node
174
Instruct new user node to connect directly to server
Go to step 102 (See Fig. 10)

Fig. 17

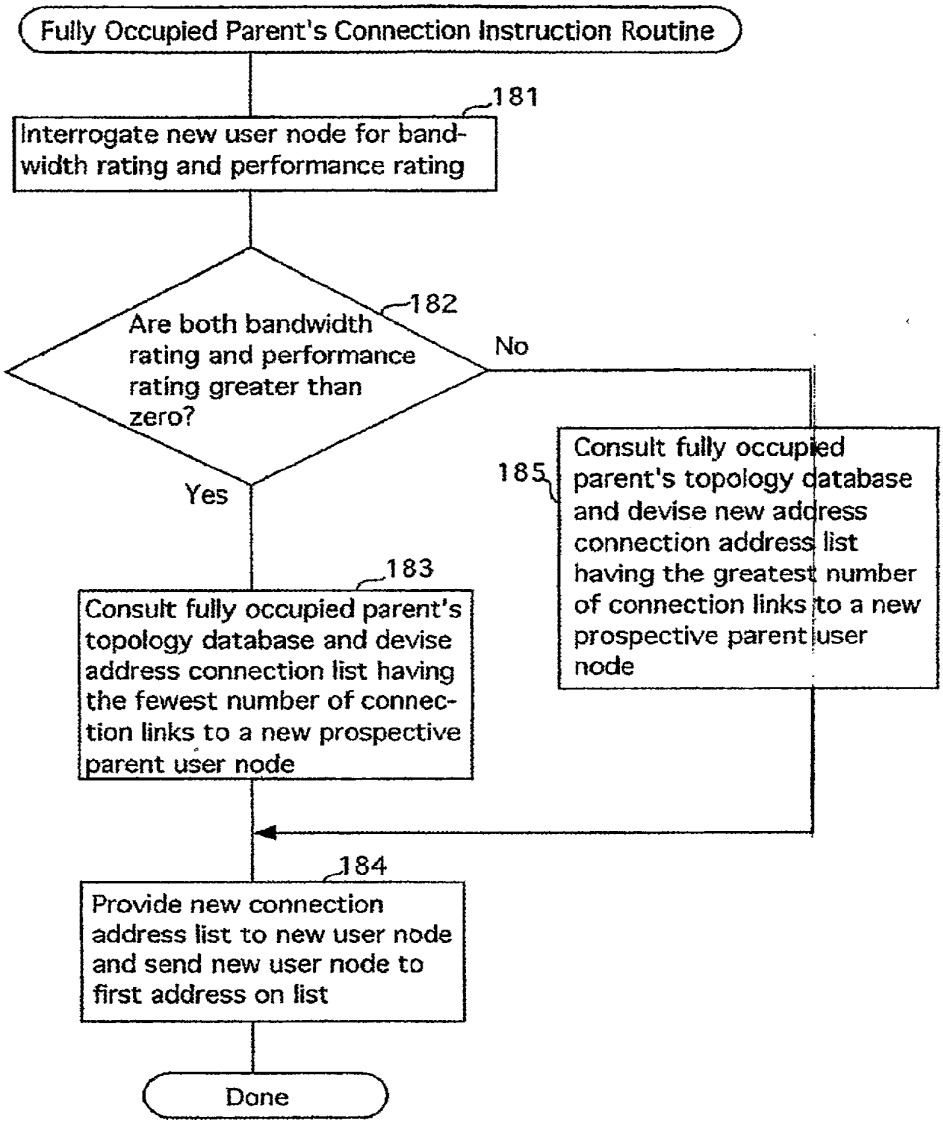

Fully Occupied Parent's Connection Instruction Routine
181
Interrogate new user node for bandwidth rating and performance rating
182
Are both bandwidth rating and performance rating greater than zero?
No
Yes
185
Consult fully occupied parent's topology database and devise new address connection address list having the greatest number of connection links to a new prospective parent user node
183
Consult fully occupied parent's topology database and devise address connection list having the fewest number of connection links to a new prospective parent user node
184
Provide new connection address list to new user node and send new user node to first address on list
Done

Fig. 18

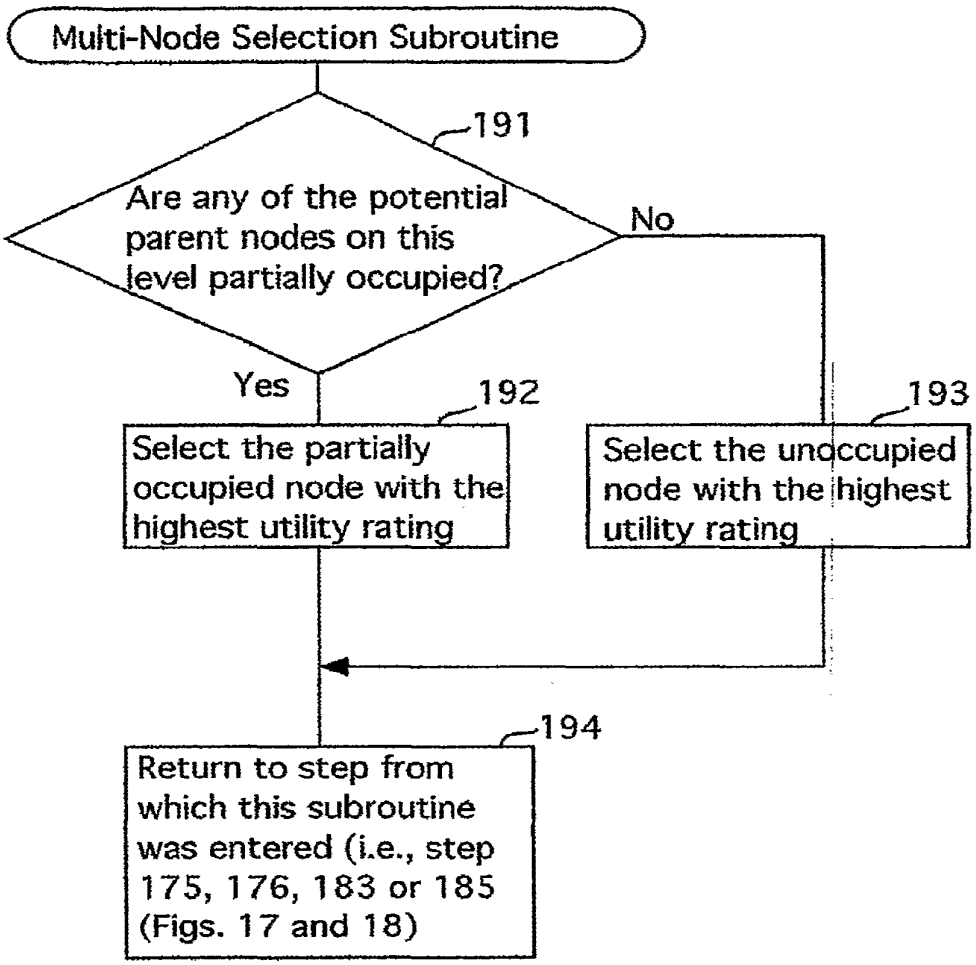

Multi-Node Selection Subroutine
191
Are any of the potential parent nodes on this level partially occupied?
No
Yes
192
Select the partially occupied node with the highest utility rating
193
Select the unoccupied node with the highest utility rating
194
Return to step from which this subroutine was entered (i.e., step 175, 176, 183 or 185 (Figs. 17 and 18)

Fig. 19

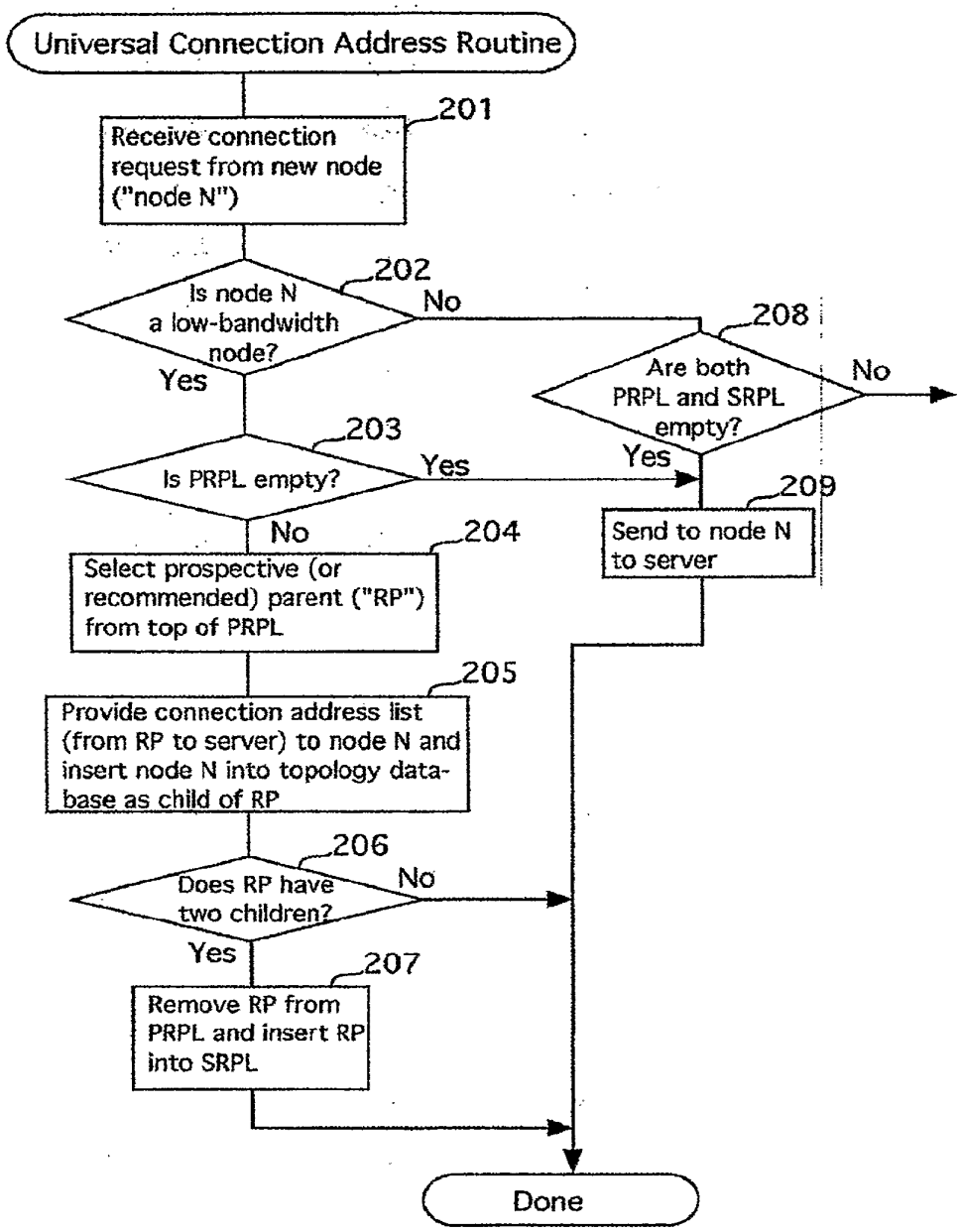

Universal Connection Address Routine
201
Receive connection request from new node ("node N")
202
Is node N a low-bandwidth node?
No
Yes
208
Are both PRPL and SRPL empty?
No
Yes
203
Is PRPL empty?
Yes
No
209
Send to node N to server
204
Select prospective (or recommended) parent ("RP") from top of PRPL
205
Provide connection address list (from RP to server) to node N and insert node N into topology data-base as child of RP
206
Does RP have two children?
No
Yes
207
Remove RP from PRPL and insert RP into SRPL
Done

Fig. 20A

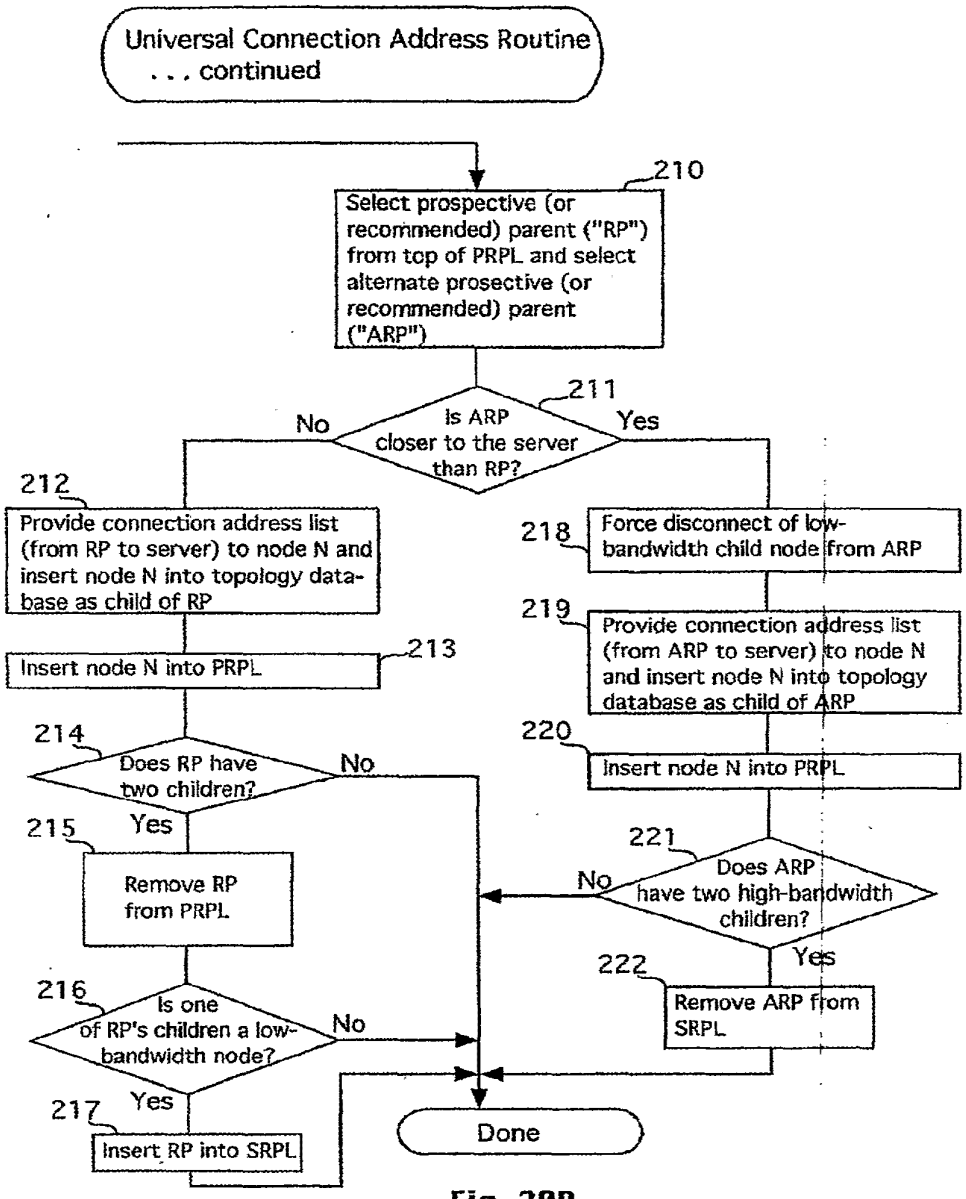

Universal Connection Address Routine
. . . continued
210
Select prospective (or recommended) parent ("RP") from top of PRPL and select alternate prosective (or recommended) parent ("ARP")
211
Is ARP closer to the server than RP?
No
Yes
212
Provide connection address list (from RP to server) to node N and insert node N into topology data-base as child of RP
213
Insert node N into PRPL
214
Does RP have two children?
No
Yes
215
Remove RP from PRPL
216
Is one of RP's children a low-bandwidth node?
No
Yes
217
Insert RP into SRPL
218
Force disconnect of low-bandwidth child node from ARP
219
Provide connection address list (from ARP to server) to node N and insert node N into topology database as child of ARP
220
Insert node N into PRPL
221
Does ARP have two high-bandwidth children?
No
Yes
222
Remove ARP from SRPL
Done

Fig. 20B

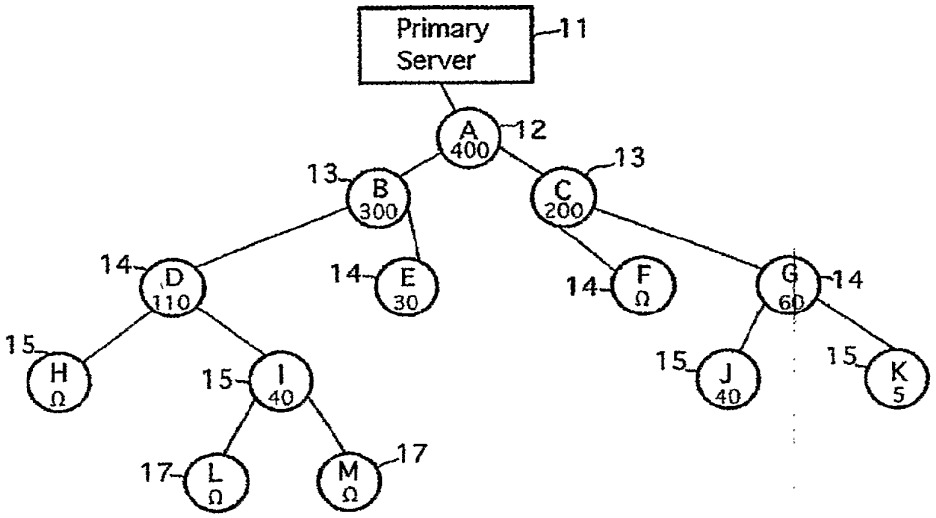

Primary Server
11
A
400
12
B
300
13
C
200
13
D
110
14
E
30
14
F
Ω
14
G
60
14
H
Ω
15
I
40
15
J
40
15
K
5
15
L
Ω
17
M
Ω
17

Fig. 21

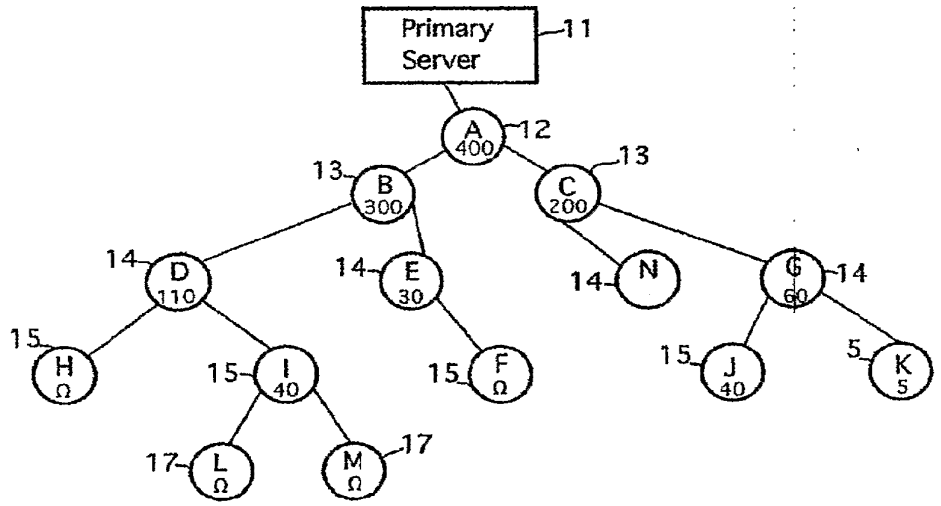

Primary Server
11
A
400
12
B
300
13
C
200
13
D
110
14
E
30
14
N
14
G
60
14
H
Ω
15
I
40
15
F
Ω
15
J
40
15
K
5
5
L
Ω
17
M
Ω
17

Fig. 22

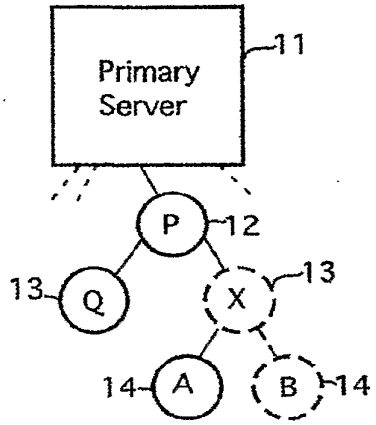


Fig. 23

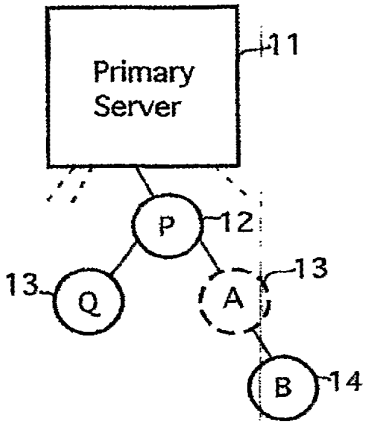


Fig. 24

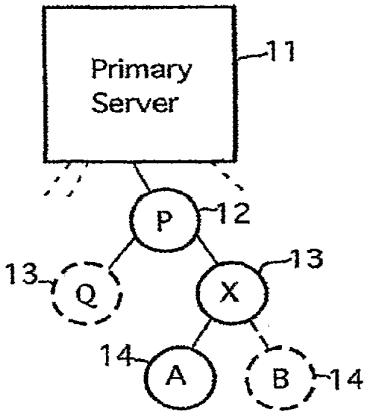


Fig. 25

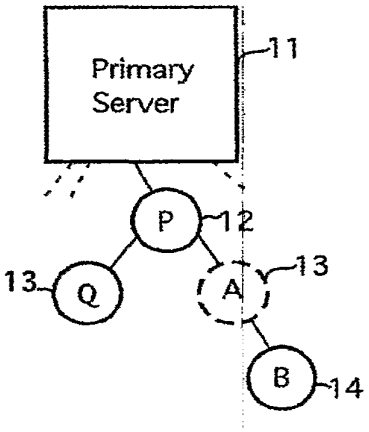


Fig. 26

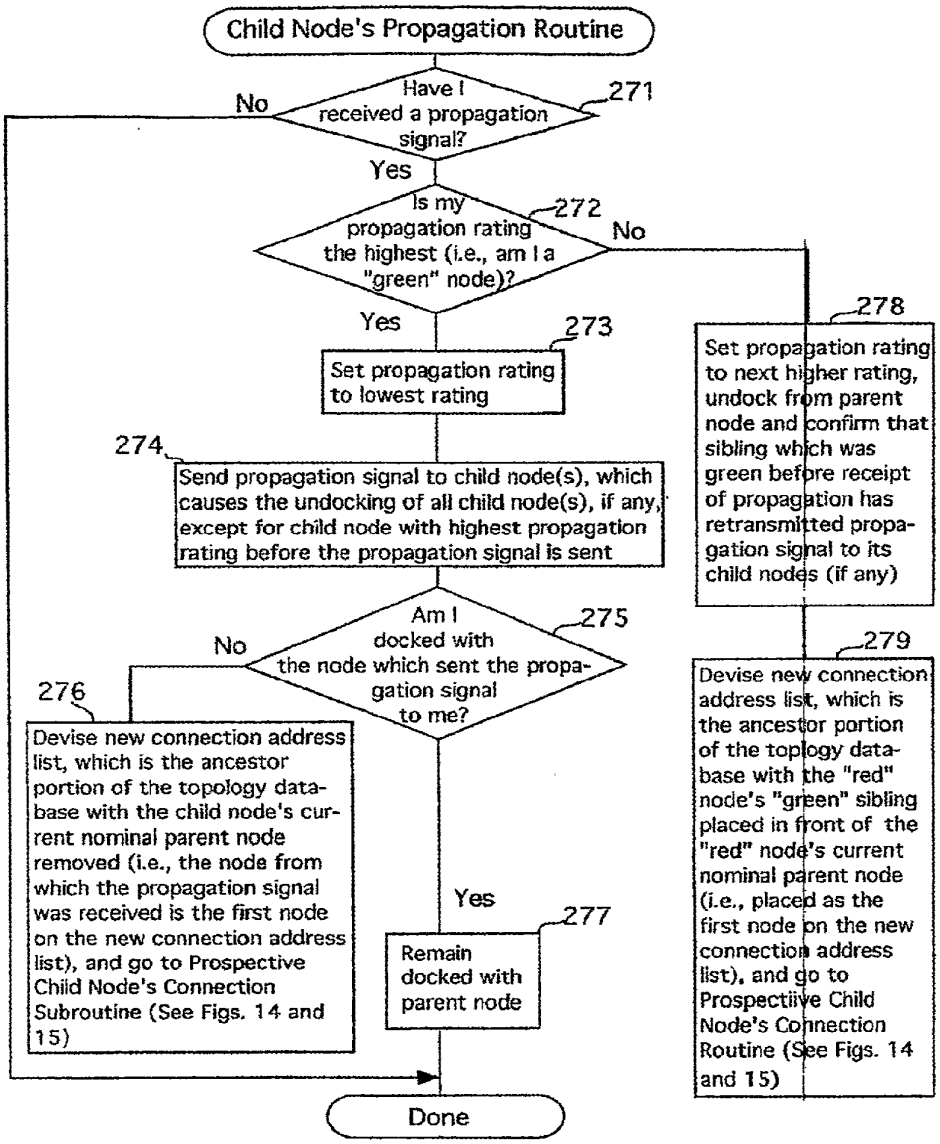

Child Node's Propagation Routine
Have I received a propagation signal?
271
No
Yes
Is my propagation rating the highest (i.e., am I a "green" node)?
272
No
Yes
273
Set propagation rating to lowest rating
274
Send propagation signal to child node(s), which causes the undocking of all child node(s), if any, except for child node with highest propagation rating before the propagation signal is sent
Am I docked with the node which sent the propagation signal to me?
275
No
Yes
276
Devise new connection address list, which is the ancestor portion of the topology database with the child node's current nominal parent node removed (i.e., the node from which the propagation signal was received is the first node on the new connection address list), and go to Prospective Child Node's Connection Subroutine (See Figs. 14 and 15)
277
Remain docked with parent node
278
Set propagation rating to next higher rating, undock from parent node and confirm that sibling which was green before receipt of propagation has retransmitted propagation signal to its child nodes (if any)
279
Devise new connection address list, which is the ancestor portion of the topology database with the "red" node's "green" sibling placed in front of the "red" node's current nominal parent node (i.e., placed as the first node on the new connection address list), and go to Prospective Child Node's Connection Routine (See Figs. 14 and 15)
Done

Fig. 27

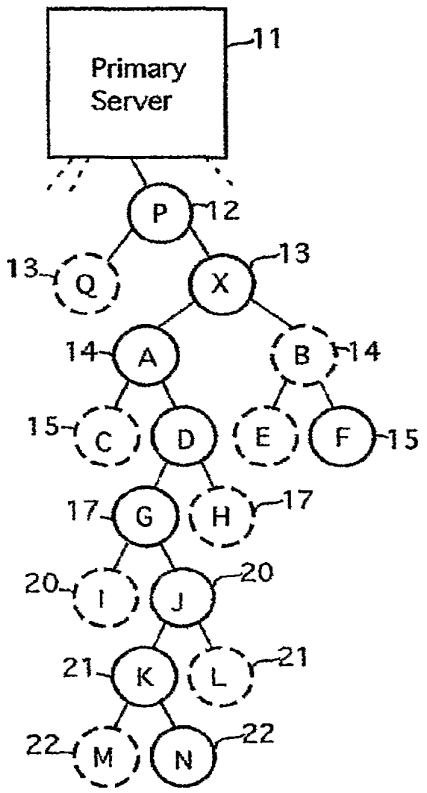

Primary Server
11
P
12
13
Q
X
13
14
A
B
14
15
C
D
E
F
15
17
G
H
17
20
I
J
20
21
K
L
21
22
M
N
22

Fig. 28

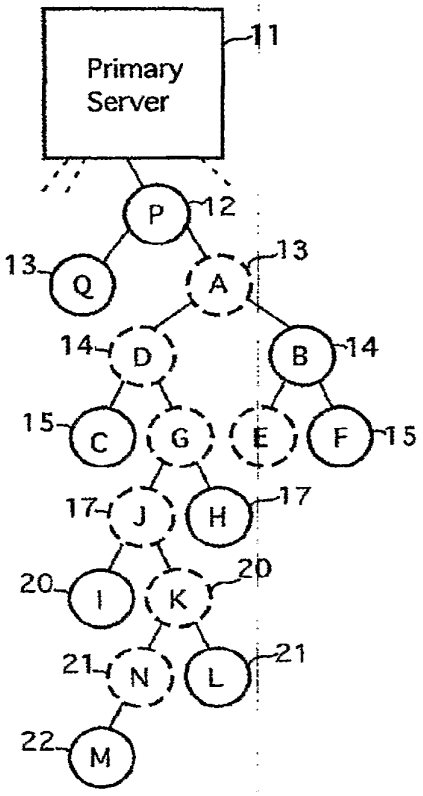

Primary Server
11
P
12
13
Q
A
13
14
D
B
14
15
C
G
E
F
15
17
J
H
17
20
I
K
20
21
N
L
21
22
M

Fig. 29

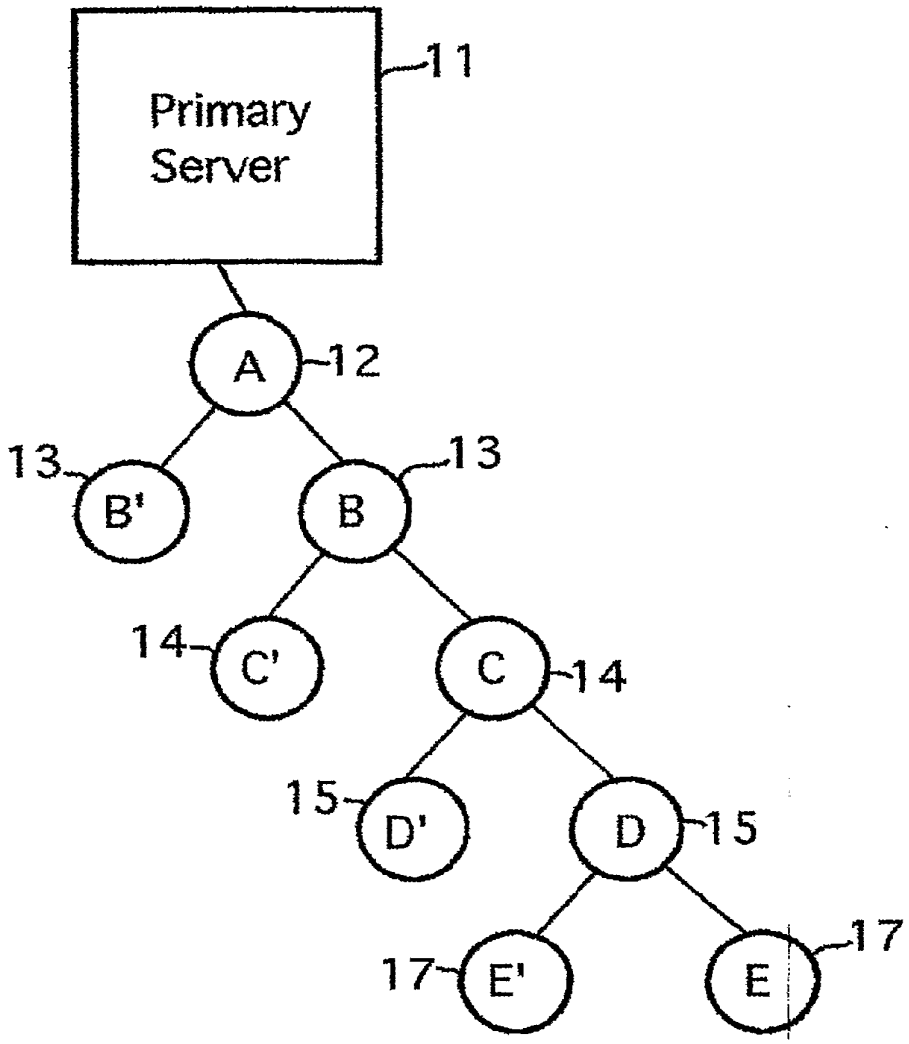


Fig. 30

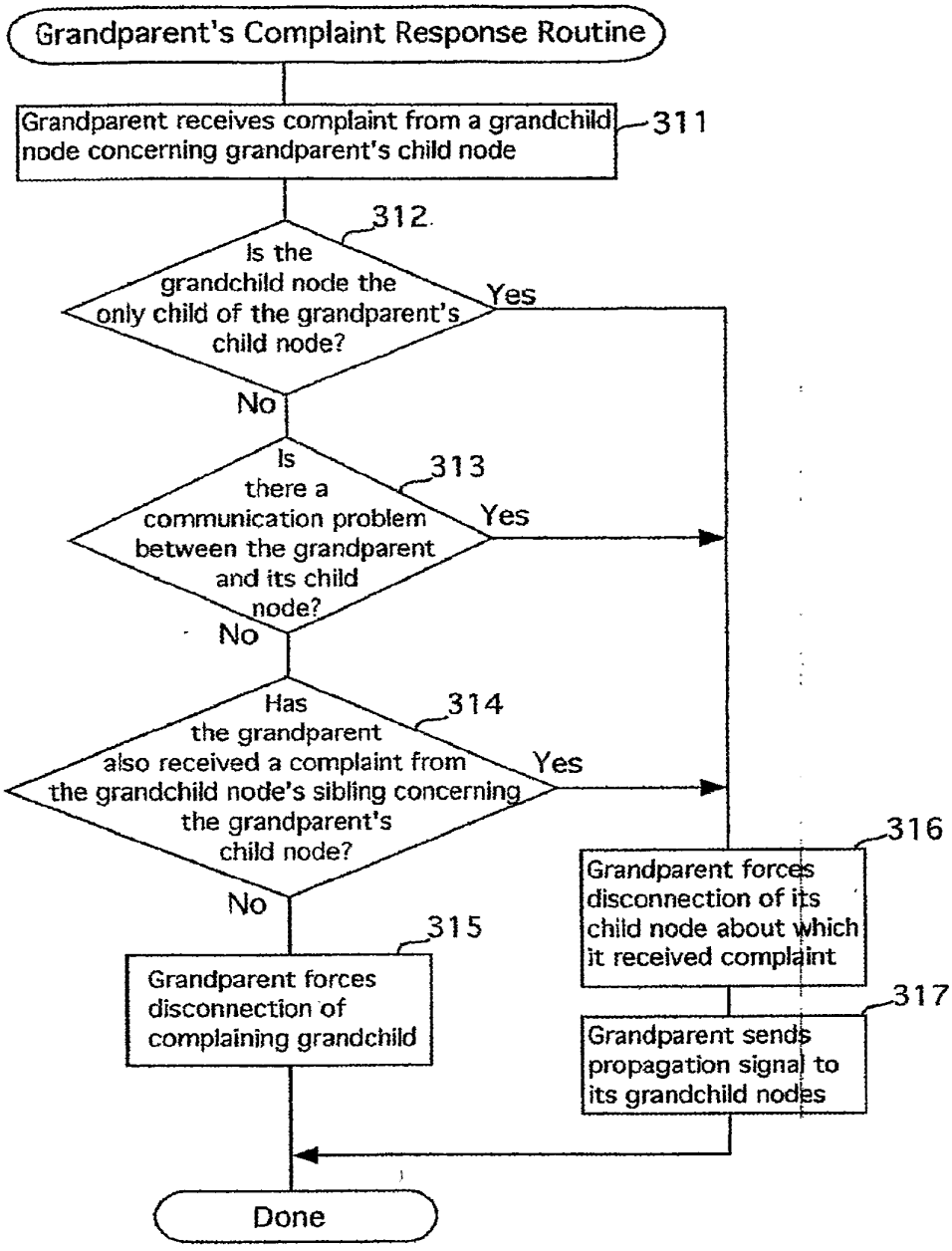

Grandparent's Complaint Response Routine
Grandparent receives complaint from a grandchild node concerning grandparent's child node
311
312
Is the grandchild node the only child of the grandparent's child node?
Yes
No
313
Is there a communication problem between the grandparent and its child node?
Yes
No
314
Has the grandparent also received a complaint from the grandchild node's sibling concerning the grandparent's child node?
Yes
No
315
Grandparent forces disconnection of complaining grandchild
316
Grandparent forces disconnection of its child node about which it received complaint
317
Grandparent sends propagation signal to its grandchild nodes
Done

Fig. 31

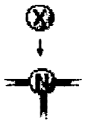


Fig. 32

Fig. 33A

Fig. 33B

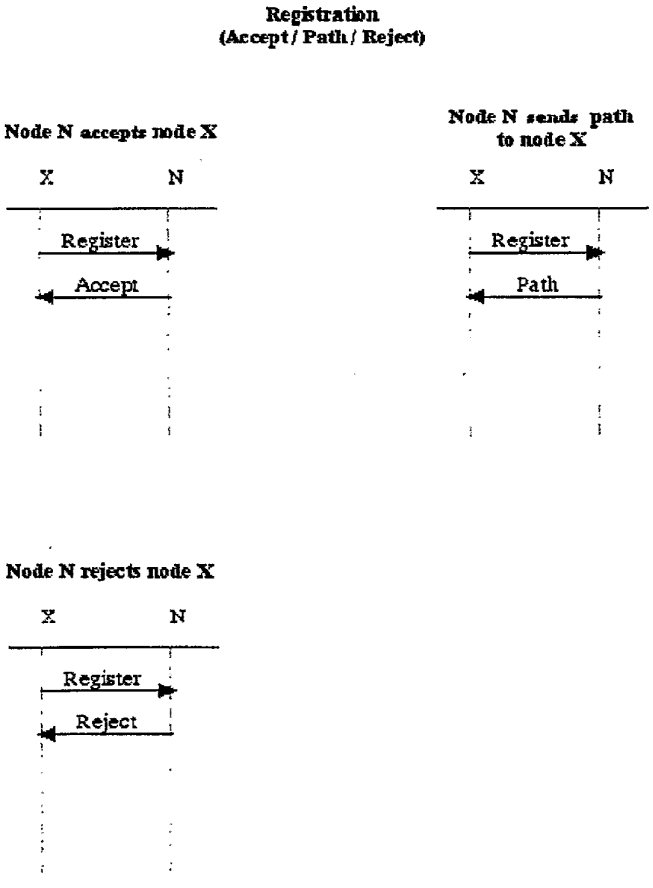


Fig. 33C

Fig. 34A

Fig. 34B

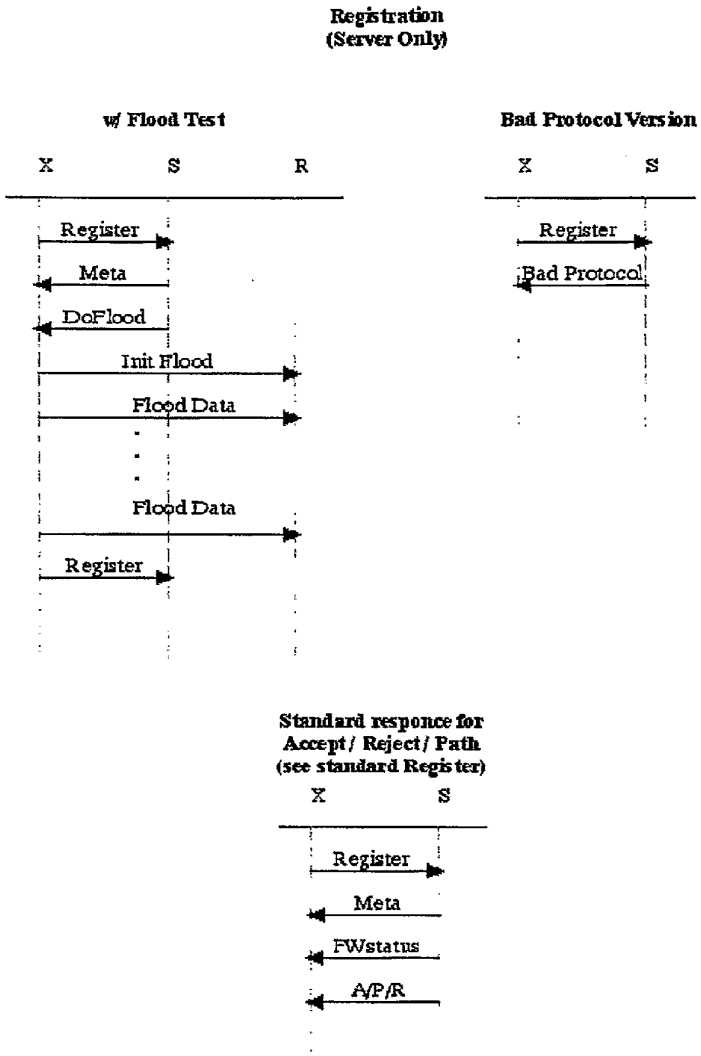


Fig. 34C

Fig. 35A

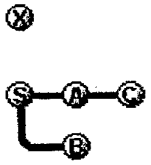
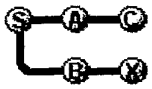


Fig. 35B



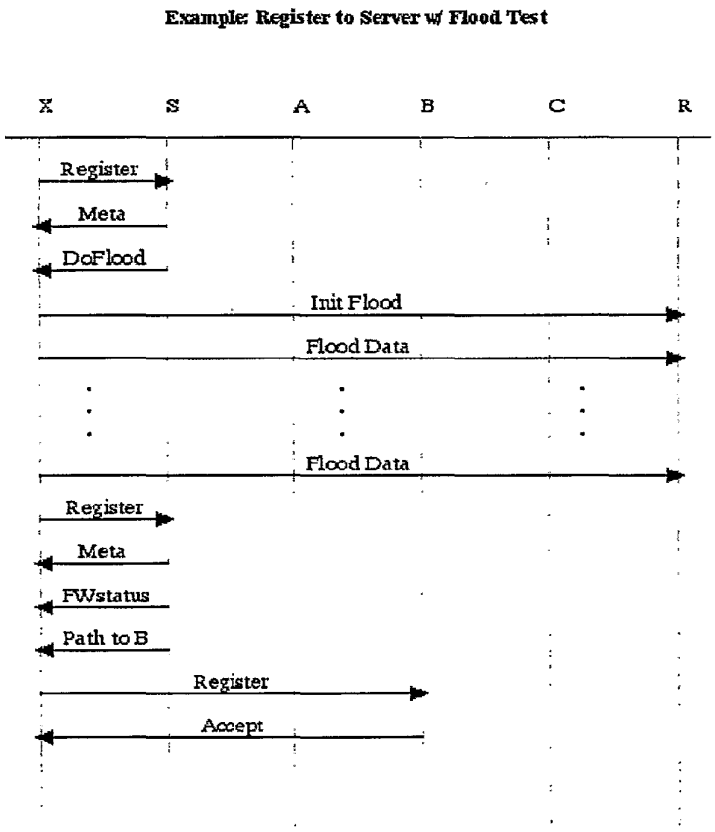


Fig. 36

Fig. 37



Fig. 38

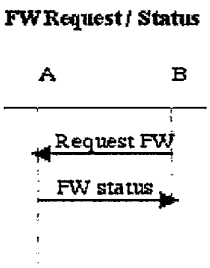


Fig. 39A



Fig. 39B

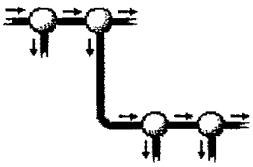
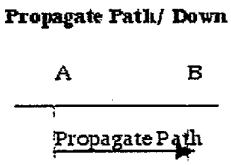


Fig. 40



28/31

Fig. 41A



Fig. 41B

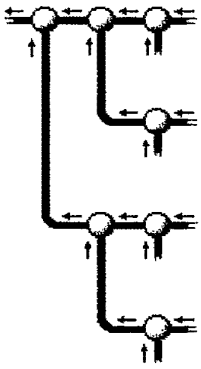


Fig. 42

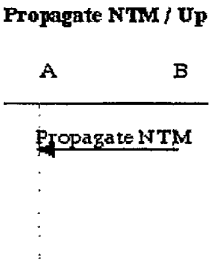


Fig. 43

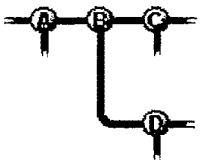


Fig. 44

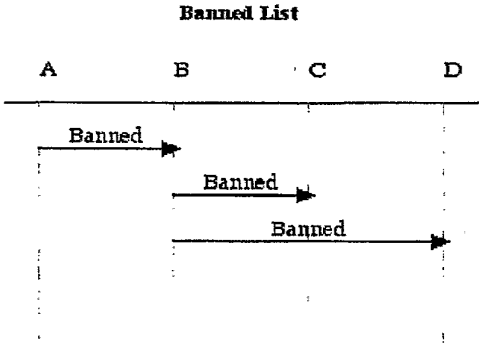


Fig. 43

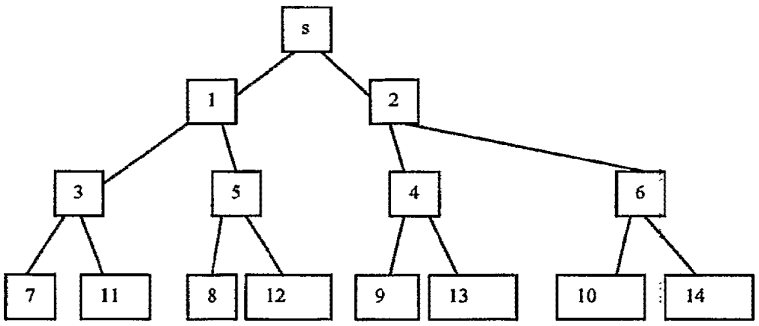


Fig. 46A

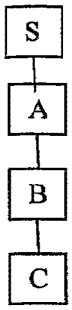


Fig. 46B

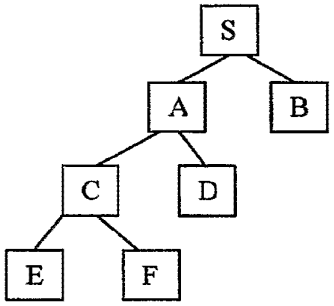
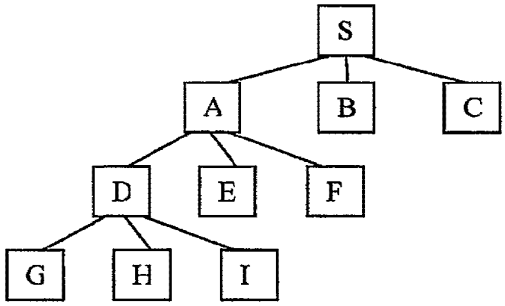


Fig. 46C



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

(11) EP 1 670 256 A2

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
14.06.2006 Bulletin 2006/24

(51) Int Cl.:
H04N 7/26 *(2006.01)*

(21) Application number: **05111156.5**

(22) Date of filing: **23.11.2005**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC NL PL PT RO SE SI SK TR
Designated Extension States:
AL BA HR MK YU

(30) Priority: **10.12.2004 US 10114**
10.12.2004 US 10113
10.12.2004 US 10040

(71) Applicant: **Microsoft Corporation**
Redmond WA 98052 (US)

(72) Inventors:
- **Klemets, Anders E.,**
  c/o Microsoft Corporation
  Redmond, WA 98052 (US)
- **Huang, Cheng,**
  c/o Microsoft Corporation
  Redmond, WA 98052 (US)
- **Chou, Philip A.,**
  c/o Microsoft Corporation
  Redmond, WA 98052 (US)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser
Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) A system and process for controlling the coding bit rate of streaming media data

(57) A system and process for controlling the coding bit rate of streaming media data is presented. This coding bit rate control involves dynamically adjusting the coding bit rate to control client buffer duration to prevent the buffer from underflowing, while keeping the average coding bit rate close to the average transmission bit rate of the network (an thus maximizing the quality of the data playback). Using the theory of optimal linear quadratic control, the client buffer duration is kept as close as possible to a target level while still keeping the coding bit rate (and hence the quality) as constant as possible. In addition, a leaky bucket model is incorporated into the control loop so that the changes in buffer duration due to natural variation in the instantaneous coding bit rate are not mistaken for changes in buffer duration due to network congestion.

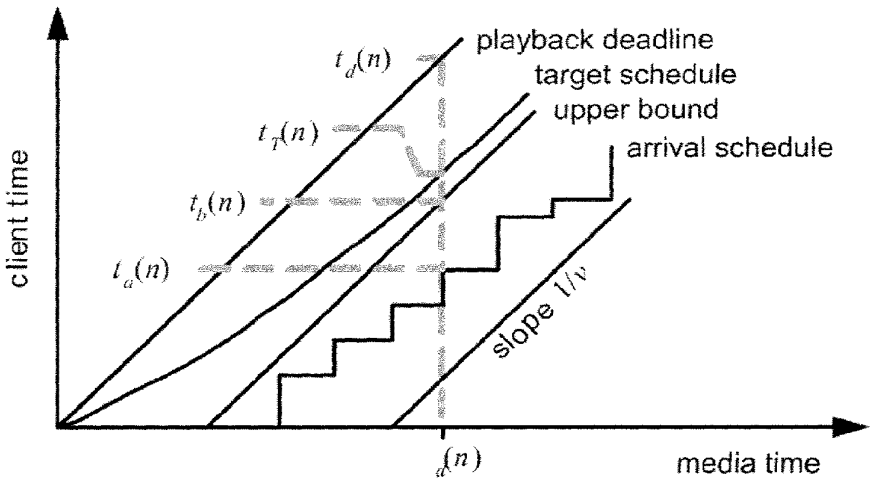


FIG. 5

EP 1 670 256 A2

Printed by Jouve, 75001 PARIS (FR)

Description

BACKGROUND

Technical Field:

[0001] The invention is related to controlling the coding bit rate of streaming media, and more particularly to a system and process for controlling the coding bit rate of streaming media data that provides fast startup, continuous playback, and maximal quality and smoothness over the entire streaming session.

Background Art:

[0002] Perhaps the major technical problem in streaming media on demand over the Internet is the need to adapt to changing network conditions. As competing communication processes begin and end, the available bandwidth, packet loss and packet delay all fluctuate. Network outages lasting many seconds can and do occur. Resource reservation and quality of service support can help, but even they cannot guarantee that network resources will be stable. If the network path contains a wireless link, for example, its capacity may be occasionally reduced by interference. Thus it is necessary for commercial-grade streaming media systems to be robust to hostile network conditions. Moreover, such robustness cannot be achieved solely by aggressive (nonreactive) transmission. Even constant bit rate transmission with re-transmissions for every packet loss cannot achieve a throughput higher than the channel capacity. Some degree of adaptivity to the network is therefore required.

[0003] End users expect that a good streaming media system will exhibit the following behavior: content played back on demand will start with low delay; once started, it will play back continuously (without stalling) unless interrupted by the user; and it will play back with the highest possible quality given the average communication bandwidth available. To meet these expectations in the face of changing network conditions, buffering of the content at the client before decoding and playback is required.

[0004] Buffering at the client serves several distinct but simultaneous purposes. First, it allows the client to compensate for short-term variations in packet transmission delay (i.e., "jitter"). Second, it gives the client time to perform packet loss recovery if needed. Third, it allows the client to continue playing back the content during lapses in network bandwidth. And finally, it allows the content to be coded with variable bit rate, which can dramatically improve overall quality. Note that even so-called constant bit rate (CBR) coded content is actually coded with variable bit rate within the constraints of a decoding buffer of a given size. The larger the decoding buffer size, the better the quality. The required decoding buffering is part of the larger client buffer.

[0005] The size of the client buffer can be expressed as the number of seconds of content in the buffer, called the buffer *duration.* The buffer duration tends to increase as content enters the buffer and tends to decrease as content leaves the buffer. Content leaves the buffer when it is played out, at a rate of v seconds of content per second of real time, where v is the *playback speed* (typically 1 for normal playback, but possibly more than 1 for high speed playback or less than 1 for low speed playback). Content enters the buffer when it arrives at the client over the network, at a rate of r_a/r_c seconds of content per second of real time, where r_a is the *arrival rate,* or average number of bits that arrive at the client per second of real time, and r_c is the *coding bit rate,* or the average number of bits needed to encode one second of content. Thus the buffer duration can be increased by increasing r_a, decreasing r_c, and/or decreasing v (and vice versa for decreasing the buffer duration). Although the buffer duration can be momentarily controlled by changing r_a or changing v, these quantities are generally not possible to control freely for long periods of time. The arrival rate r_a on average is determined by the network capacity, while the playback speed v on average is determined by user preference. Thus if the network capacity drops dramatically for a sustained period, reducing the coding bit rate r_c is the only appropriate way to prevent a *rebuffering event* in which playback stops (v=0) while the buffer refills.

[0006] Thus, adaptivity to changing network conditions requires not only a buffer, but also some means to adjust the coding bit rate r_c of the content. This can be done by stream switching in combination with multi bit rate (MBR) coding or coarse grained or fine grained scalable coding. Today's commercial streaming media systems [1] rely on MBR coding as well as *thinning,* which is a form of coarse grained scalability. In MBR coding, semantically identical content is encoded into alternative bit streams at different coding bit rates and stored in the same media file at the server, allowing the content to be streamed at different levels of quality corresponding to the coding bit rates r_c, possibly using bit stream switching [2]. In coarse grained scalable coding (such as MPEG-2/4 temporal or SNR scalability) the content is encoded into several sub-streams or *layers,* so that the coding bit rate r_c can be changed in large deltas by adding or dropping (at possibly restricted times) one layer of content at a time. Thinning is a special case of coarse grained scalability in which dependent video frames (P and B frames) are dropped before independent video frames (I frames), which are in turn are dropped before audio frames. Future commercial systems may support fine grained scalability (FGS) as well. Fine grained scalable coding (such as 3D SPIHT [6], MPEG-4 FGS [7], or EAC [8]) allows the coding bit rate r_c to change

Aylo Exhibit 1002
Page 216 of 1405

at any time in deltas sometimes as small as one byte per presentation. FGS coding offers great flexibility in adapting to variable network conditions, and can demonstrably improve quality under such conditions.

[0007] Some examples of existing technology that adjusts the coding bit rate r_c of the content in an attempt to adapt to changing network conditions includes de Cuetos and Ross [9], which decouples the transmission rate and the coding bit rate. They assume that the transmission rate is determined by the network transport protocol (TCP or TFRC). Based on this, they develop a heuristic real time algorithm for adaptive coding bit rate control and compare its performance to an optimal offline coding bit rate control policy if the transmission rate is given prior to streaming. The work of Rejaie, Handley and Estrin [4] proposes a scheme for transmitting layered video in the context of unicast congestion control, which basically includes two mechanisms. One mechanism is a coarse-grained mechanism for adding and dropping layers (changing the overall coding bit rate and quality). The other is a fine-grained interlayer bandwidth allocation mechanism to manage the receiver buffer (not changing the overall coding bit rate or quality). A potential issue with this approach is that it changes the coding bit rate by adding or dropping one (presumably coarse) layer at a time. If the layers are fine-grained, as in the case of FGS coded media, then adding or dropping one (fine-grained) layer at a time typically cannot provide a prompt enough change in coding bit rate. Moreover, since the adding and dropping mechanism is rather empirical, the mechanism may simply not be suitable for FGS media. The work of Q. Zhang, Zhu and Y-Q. Zhang [5] proposes a resource allocation scheme to adapt the coding bit rate to estimated network bandwidth. The novelty of their approach is that they consider minimizing the distortion (or equivalently maximizing the quality) of all applications, such as file-transfers and web browsing in addition to audio/video streaming. However, their optimization process does not include the smoothness of individual streams and might lead to potential quality fluctuations.

[0008] However, even with buffering and the ability to adjust the coding bit rate, existing technologies for streaming media on demand over the Internet suffer from two problems:

1. Playback often stalls during network congestion. That is, during playback of high bit rate content, if the network bit rate drops below the content bit rate, the client buffer runs out of content and playback stops while the client rebuffers (known as a "rebuffering" event).
2. Start-up delay is often too long (about 5 seconds).

There are existing solutions to both of these problems, but they do not always work well. One solution to the first problem is to stream the content encoded at a coding bit rate that is low relative to the average bit rate transmitted over the network (the *transmission bit rate*). This will enable the buffer to build up over time. With such a large reserve of unplayed information on the client, temporary network congestion will not affect playback. However, this solution has two problems. First, the coding bit rate of the content is not as high as the average transmission bit rate of the network and hence the quality is lower than it could be. Second, the buffer can grow nearly as large as the streamed file itself. This may demand too many resources on the client device.

[0009] Another solution to the first problem is to try to maintain the client buffer at a constant level (typically about 10 seconds), while switching between different coding bit rates for the same content, trying to match the transmission bit rate of the network. However, rebuffering events are still commonly observed in practice, because choosing the right time to switch streams is difficult. One reason that it is difficult is that there are natural variations in the instantaneous coding bit rate of the content, even in so-called constant bit rate encodings, which can confuse the client buffer management algorithm.

[0010] The second problem above (long start-up delay) also has multiple solutions. One solution is to fill up the client buffer quickly, with a quick initial transmission rate burst. With the client buffer full, playback can safely begin. However, this solution has several problems. First, it is only applicable when there is sufficient "headroom" in the network to increase the transmission bit rate for a few seconds. Thus it is usually not applicable for modem connections, for example. Second, it stresses the network, causing other applications in the network to back off. It has been shown that during the burst period, there can be as much as 80% packet loss, causing all TCP connections sharing the same bottleneck to back off. Third, by implication, if there is headroom in the network for bursting, then the streaming application may not be using the full bandwidth available to it during the remainder of the file, meaning that quality is lower than it should be.

[0011] Another solution to the second problem is to play back the content slower than real time, allowing playback to begin while the client buffer builds up. This is an innovative solution, but has the obvious temporal distortion.

[0012] A final solution to the second problem is to lower temporarily the coding bit rate of the content below the transmission bit rate of the network, allowing playback to begin while the client buffer builds up. This is a solution proposed by Chou et al. in [13].

[0013] The system and process of the present invention resolve the problems of the existing techniques and provide fast startup, continuous playback, and maximal quality and smoothness over the entire streaming session.

[0014] It is noted that in the preceding paragraphs, as well as in the remainder of this specification, the description refers to various individual publications identified by a numeric designator contained within a pair of brackets. For example, such a reference may be identified by reciting, "reference [1]" or simply "[1]". A listing of references including

Aylo Exhibit 1002
Page 217 of 1405

the publications corresponding to each designator can be found at the end of the Detailed Description section.

SUMMARY

[0015] The present invention is directed toward a system and process for controlling a coding bit rate of streaming media data being transmitted to a client from a server over a computer network. In general, this coding bit rate control involves dynamically adjusting the coding bit rate of the streaming media data to control the client buffer duration. The purpose of this is to prevent the client buffer from underflowing, while keeping the average coding bit rate close to the average transmission bit rate of the network (an thus maximizing the quality of the playback of the data). The problem of coding bit rate control is formulated as a standard problem in linear quadratic optimal control, in which the client buffer duration is controlled as closely as possible to a target level. The smoothness of the average coding bit rate over consecutive frames is also considered when deciding whether to change the coding bit rate as part of the optimal control process. This yields a higher and more stable quality as network conditions change. In addition, the natural variation in the instantaneous coding bit rate that occurs for a given average coding bit rate is explicitly take into consideration. This is accomplished by incorporating the leaky bucket model into the control loop so that the changes in buffer duration due to natural variation in the instantaneous coding bit rate are not mistaken for changes in buffer duration due to network congestion. It is noted that in the present system and process, it is not the actual fullness of the client buffer that is controlled, but an upper bound on the time of arrival of bits into the client buffer, to a target level. The upper bound is based on the leaky bucket model of the coding bit rate.

[0016] Preventing the client buffer from underflowing, while keeping the average coding bit rate close to the average transmission bit rate of the network, meets two of the aforementioned user expectations. Namely, if the buffer never underflows, it allows for continuous playback. In addition, keeping the average coding bit rate close to the average transmission bit rate of the network means the data is played back with the highest possible quality given the average communication bandwidth available. This leaves the remaining issue of startup delay. This issue is resolved in the present system and process by controlling the size of the client buffer over time. More particularly, the aforementioned client buffer target level is adjusted to start small, and then grow slowly over time. If the buffer is initially small, it allows for shorter startup delay. In addition, as the buffer is eventually allowed to grow large, it enhances the robustness of the system as well as creating high, nearly constant quality. Thus, client buffer management is a key element affecting the performance of streaming media systems.

[0017] More particularly, the present system and process involves a server generating a streaming media data stream that exhibits one of a number of coding bit rates supported by the server. Initially, the server chooses the coding bit rate during a startup period. However, after the startup period, the client provides coding bit rate requests to the server. In response, the server transmits the streaming media data at the most appropriate supported coding bit rate closest to the rate requested by the client. The coding bit rates requested by the client are those estimated to provide a high quality playback of the streaming media data while still keeping a decoder buffer of the client used to receive streaming media data from the server filled to a desired duration level.

[0018] The client computes the coding bit rate that will provide the desired results on an ongoing basis using the aforementioned linear quadratic optimal control technique. This coding bit rate computation involves determining on a frame-by-frame basis a coding bit rate that reduces a difference between an estimated latest anticipated arrival time of the frame under consideration and a prescribed target arrival time, while at the same time reducing the change in the coding bit rate to a prescribed degree. In one embodiment involving the aforementioned leaky bucket model, the coding bit rate computations are based on parameters indicative of the state of the encoder buffer of the server. These parameters are computed by the server and provided to the client along with the streaming media data.

[0019] More particularly, the server first computes a set of parameters defining an initial state of the encoder buffer as it would exist if a data stream corresponding to a supported coding bit rate where streamed therefrom. A separate set of parameters is computed for each of the supported coding bit rates, and are provided to the client in a preamble to the streaming media data. These parameters include the coding bit rate associated with the data stream, the size of the encoder buffer employed with the coding bit rate of the data stream, and a value indicative of the initial encoder buffer fullness exhibited at the coding bit rate of the data stream. It is noted that the size of the encoder buffer employed with each supported coding bit rate varies and is chosen so as to be the minimum size buffer that will still contain the data stream at any point in the streaming process given the coding bit rate and the initial encoder buffer fullness.

[0020] In addition to the initial encoder buffer parameters, the server also computes an upper bound gap for each frame of the streaming media data generated for each coding bit rate supported by the server. This upper bound gap is defined as the number of data bits that the server's encoder buffer can contain over the bits currently contained therein after a just-generated frame is fully input into the buffer. More particularly, the upper bound gap for each frame is computed as the difference between the encoder buffer size and the last computed value for an encoder buffer fullness after insertion of the just-generated frame. The encoder buffer fullness value after insertion of the just-generated frame is computed as the sum of the last computed value of the encoder buffer fullness value prior to insertion of the just-

Aylo Exhibit 1002
Page 218 of 1405

generated frame and the size of that just-generated frame. The encoder buffer fullness value prior to insertion of the just-generated frame is computed as either zero, or the difference between the value computed for the encoder buffer fullness after the insertion of the frame generated immediately before the just-generated frame and the coding bit rate associated with that prior frame divided by the instantaneous frame rate associated with said prior frame, whichever is larger. The instantaneous frame rate is equal to the reciprocal of, the time the next frame is scheduled to be encoded less the time the just-generated frame was encoded. In one embodiment, the server provides the upper bound gap computed for the first frame of a sequence of frames generated after a change in the coding rate to the client along with an indication of the coding bit rate associated with the sequence of frames as part of the data associated with the first frame of the sequence. In another embodiment, the server provides the gap value with every frame generated.

[0021] The client employs the encoder buffer parameters and the upper bound gap values to determine on a frame-by-frame basis the coding bit rate that will reduce a difference between the estimated latest anticipated arrival time of a frame under consideration and a prescribed target arrival time of that frame, in order to keep the client buffer at approximately the desired duration level. In cases where the upper bound gap is only provided with the first frame after a change in the coding bit rate, the client estimates the upper bound gap for the frames that do not include a gap value. It is also noted that given the aforementioned initial encoder buffer conditions, it is possible for the client to estimate the upper bound gap values for every frame received. Thus, an alternate embodiment where the server is not configured to compute and provide the gap values on an ongoing basis, the client computer can still compute them on its own.

[0022] When a frame with a new average coding bit rate arrives at the client there is a shift in the upper bound gap. This shift can be on the order of seconds and hence, rather than being negligible, can be confusing to the controller. One solution to this is to introduce a simultaneous shift in the control target schedule. To this end, the server also computes a shift value for each frame representing the first frame after a coding bit rate change. This shift value represents the difference between the upper bound gap that would be associated with the frame generated immediately before this first frame had it been encoded at the new coding bit rate and the upper bound gap actually associated with the frame as encoded at the previous coding bit rate. The shift value is provided to the client along with the first frame after a change in the coding bit rate. The client shifts the currently scheduled target arrival time associated with the just received "first" frame by the shift value provided, and the currently scheduled target arrival times for future frames are shifted such that they collectively approach, over a period of time, and eventually coincide with, the previous target arrival times for those frames. In this way the adverse effects of an upper bound gap shift when the coding bit rate is changed are mitigated and the target arrival time values are eventually brought back in line.

[0023] In regard to the target arrival time for each frame, these are chosen so as to make the amount of time between the target time of a frame and its playback time large enough, after the startup period, that network jitter, delays and throughput changes which may cause the actual arrival time of the frame to be after its target arrival time do not result in the frame arriving after its scheduled playback time. During the startup period, the target arrival times are chosen so as to make the target arrival time closer to the playback time to assist in reducing startup delay. In one embodiment this is accomplished by setting the target arrival time for each frame using a logarithmic target schedule. In another embodiment this is accomplished by setting the target arrival time for each frame using a two-piece linear target schedule where the difference between the target arrival time and the playback time for a frame arriving during the startup period increases linearly to a prescribed amount of time that is large enough to account for the network jitter, delays and throughput changes and after which the difference remains substantially constant.

[0024] It is noted that the startup delay can be further minimized by beginning playback of the first frame of the streaming media data when the client's decoder buffer has the minimum amount of data therein that is needed to ensure an underflow condition does not occur at the initial coding bit rate. Still further, the initial coding bit rate can be set to a level that is less than the anticipated arrival rate of the streaming media data at the client since the minimum amount of data needed to ensure an underflow condition does not occur, decreases as the arrival rate increases relative to the coding bit rate. Thus, in the startup period before the client request changes in the coding bit rate, the server provides the streaming media data at the initial coding bit rate. In one embodiment, the initial coding bit rate is set to approximately one-half the anticipated arrival rate, where the anticipated arrival rate is estimated to be the anticipated transmission rate from the server at the current bandwidth available on the network. It is noted that the startup period can be defined as the period of time prior to the first instance of an estimated latest arrival time of a frame under consideration computed for the initial coding bit rate being earlier than a target arrival time for that frame.

[0025] In regard to the client identifying the coding bit rate for each frame that also reduces the change in the coding bit rate to a prescribed degree, it is noted that the prescribed degree can vary depending on whether the coding bit rate increased. If it did increase, then the prescribed degree to which any future change in the coding bit rate is minimized is made greater in comparison to the prescribed degree in cases where the last change in the coding bit rate decreased the rate. Still further, it is also desirable to minimize quality variations due to large or frequent changes to the coding bit rate. In order to stabilize the code bit rate changes, the following actions can be taken. First, in the case where the client identifies a new coding bit rate that represents an increase, a new coding bit rate would be requested from the server only if the new rate does not exceed the current moving average arrival rate of the frames. In another embodiment, the

Aylo Exhibit 1002
Page 219 of 1405

new coding bit rate representing an increase would only be requested from the server, even if it exceeds the current moving average arrival rate of the frames, if the current client buffer duration exceeds the desired duration level by an amount that it is estimated will not be expended at the higher coding bit rate prior to the passing of a prescribed period of time (e.g., 60 seconds). However, in a case where the client identifies a new coding bit rate that represents a decrease, a new coding bit rate would be requested from the server even if it represents a significant departure from the immediately previous coding bit rate. This is because there is little risk in the client buffer underflowing when the coding rate is decreased.

[0026] In regard to the aforementioned moving average arrival rate of the frames, a new procedure for computing this rate is employed with the present system and process. More particularly, the moving average arrival rate is computed on a packet-by-packet basis, by calculating the product of the moving average arrival rate computed for the immediately preceding packet to a currently received packet and a fractional weighting factor, added to the product of the instantaneous arrival rate of the currently received packet and one minus the fractional weighting factor. In this computation, the fractional weighting factor is not a constant as in the past, but instead is based on the inter-arrival gaps between received packets. More particularly, the fractional weighting factor $\beta(k)$ for a currently received packet k is computed as $\frac{e^{-\alpha[t(k)-t(k-1)]}-e^{-\alpha[t(k)-t(0)]}}{1-e^{-\alpha[t(k)-t(0)]}}$ where α is the reciprocal of a prescribed time constant, $t(k)$ is the actual arrival time of the current packet, $t(k\text{-}1)$ is the actual arrival time of the packet received immediately prior to the current packet, and $t(0)$ is the arrival time of the first packet of the streaming media data. In this case, the instantaneous arrival rate $r_a(k)$ of the current packet k is computed as $\frac{b(k)}{t_a(k)-t_a(k-1)}$, where $b(k)$ is the size of the current packet. If $b(k)$ is expressed in bits and $t_a(k) - t_a(k - 1)$ is expressed in seconds, then the moving average arrival rate represents the arrival rate of the streaming media data bit bits per second. It is noted that a similar procedure can be employed for any arbitrary units of rate of the streaming media.

[0027] As mentioned previously, the server generates data streams that exhibit one of a number of coding bit rates supported by the server, and transmits data streams at the most appropriate supported coding bit rate closest to the rate requested by the client. The number of available coding bit rates will depend on how the data is encoded. For example, if a fine grain scalable coding scheme is used theoretically there could be a large number of rates available (although for practical reasons the number is more likely to be lower, e.g., 50). However, if a coarse grain scalable coding scheme or a multiple bit rate coding scheme is employed, there could be a more limited number of coding bit rates available from the server. Thus, in some cases an optimum coding bit rate identified by the client may not be available from the server. In addition, even if there is a matching coding bit rate available, the upper bound gap may be such that switching to that rate would risk a client buffer underflow. Given this, the present system and process includes techniques for determining the most appropriate coding bit rate, from the available rates, in view of the optimum coding bit rate identified by the client. It is noted that the available rates could be provided to the client from the server in a preamble to the streaming media data. In such a case, the client itself would perform the analysis and request the resulting supported coding bit rate from the server. However, in an alternate embodiment, the client would request the optimum coding bit rate it identified, and the server would perform the analysis to determine which supported rate is most appropriate. The server would then provide a data stream exhibiting the selected coding bit rate. In either case, the analysis involves finding a supported coding bit rate that is equal to, or if none are equal, the closest smaller rate to, the optimum coding bit rate identified by the client. Whenever the supported coding bit rate found is lower than the coding bit rate associated with the last generated frame of the streaming media data, all future frames (or those beginning with a frame specified by the client) are generated at that supported rate. However, when the supported coding bit rate found is higher than the coding bit rate associated with the last generated frame, it is determined if a difference between the upper bound gap associated with the last generated frame had it been encoded at the supported coding bit rate found and the upper bound gap associated with that frame encoded at the current coding bit rate, is less than or equal to a maximum allowable difference value. If the difference is less than the maximum allowable difference value, the affected future frames are generated at the supported rate found. But, if the difference is not less than the maximum allowable difference value, the next lower supported coding bit rate is found and the foregoing actions are repeated to ultimately identify the appropriate rate.

[0028] The maximum allowable difference value is computed by the client, and if the server is doing the analysis, the client provides the value along with the request for data at the new coding bit rate. The client chooses the maximum allowable difference value such that the latest anticipated arrival time associated with a frame provided by the server immediately prior to a frame under consideration by the client, had it been coded at the requested coding bit rate, is no more than a prescribed fraction of the way from that frame's target arrival time to its playback deadline (e.g., 1/3 of the way).

[0029] In addition to the just described benefits, other advantages of the present invention will become apparent from

Aylo Exhibit 1002
Page 220 of 1405

the detailed description which follows hereinafter when taken in conjunction with the drawing Fig.s which accompany it.

DESCRIPTION OF THE DRAWINGS

[0030] The specific features, aspects, and advantages of the present invention will become better understood with regard to the following description, appended claims, and accompanying drawings where:

FIG. 1 is a diagram depicting a general purpose computing device constituting an exemplary system for implementing the present invention.

FIG. 2 is block diagram of a simplified streaming media communication pipeline.

FIG. 3 is a graph showing schedules at which bits in the coded bit stream pass points A, B, C and D in the communication pipeline of Fig. 2 in terms of bits vs. media time.

FIG. 4 is a graph showing a buffer tube containing a coding schedule in terms of bits vs. media time.

FIG. 5 is a graph showing an arrival schedule and its upper bound in client time vs. media time, where the upper bound is controlled to a target schedule that is increasingly in advance of the playback deadline to provide greater robustness over time.

FIG. 6 is a graph showing a target arrival schedule design.

FIGS. 7(a)-(b) are graphs showing two different target arrival schedules. Fig. 7(a) shows a logarithmic schedule and Fig. 7(b) shows a two-piece linear schedule.

FIG. 8 is a graph showing buffer tubes for various transmission rates.

FIG. 9 is a graph illustrating exponential averaging.

FIG. 10 is a graph showing buffer tube changes and control target adjustments.

FIGS. 11A-C show a flow chart diagramming an embodiment of the coding bit rate control process according to the present invention.

FIG. 12 is a timeline illustrating the conservative limit used in the conservative coding bit rate up-switching procedure.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0031] In the following description of the preferred embodiments of the present invention, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

I. The Computing Environment

[0032] Before providing a description of the preferred embodiments of the present invention, a brief, general description of a suitable computing environment in which portions of the invention may be implemented will be described. Fig. 1 illustrates an example of a suitable computing system environment 100. The computing system environment 100 is only one example of a suitable computing environment and is not intended to suggest any limitation as to the scope of use or functionality of the invention. Neither should the computing environment 100 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in the exemplary operating environment 100.

[0033] The invention is operational with numerous other general purpose or special purpose computing system environments or configurations. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

Aylo Exhibit 1002
Page 221 of 1405

[0034] The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote computer storage media including memory storage devices.

[0035] With reference to Fig. 1, an exemplary system for implementing the invention includes a general purpose computing device in the form of a computer 110. Components of computer 110 may include, but are not limited to, a processing unit 120, a system memory 130, and a system bus 121 that couples various system components including the system memory to the processing unit 120. The system bus 121 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0036] Computer 110 typically includes a variety of computer readable media. Computer readable media can be any available media that can be accessed by computer 110 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer readable media may comprise computer storage media and communication media. Computer storage media includes both volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 110. Communication media typically embodies computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer readable media.

[0037] The system memory 130 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 131 and random access memory (RAM) 132. A basic input/output system 133 (BIOS), containing the basic routines that help to transfer information between elements within computer 110, such as during start-up, is typically stored in ROM 131. RAM 132 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 120. By way of example, and not limitation, Fig. 1 illustrates operating system 134, application programs 135, other program modules 136, and program data 137.

[0038] The computer 110 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Fig. 1 illustrates a hard disk drive 141 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 151 that reads from or writes to a removable, nonvolatile magnetic disk 152, and an optical disk drive 155 that reads from or writes to a removable, nonvolatile optical disk 156 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 141 is typically connected to the system bus 121 through a non-removable memory interface such as interface 140, and magnetic disk drive 151 and optical disk drive 155 are typically connected to the system bus 121 by a removable memory interface, such as interface 150.

[0039] The drives and their associated computer storage media discussed above and illustrated in Fig. 1, provide storage of computer readable instructions, data structures, program modules and other data for the computer 110. In Fig. 1, for example, hard disk drive 141 is illustrated as storing operating system 144, application programs 145, other program modules 146, and program data 147. Note that these components can either be the same as or different from operating system 134, application programs 135, other program modules 136, and program data 137. Operating system 144, application programs 145, other program modules 146, and program data 147 are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information into the computer 110 through input devices such as a keyboard 162 and pointing device 161, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 120 through a user input interface 160 that is coupled to the system bus 121, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB). A monitor 191 or other type of display device is also connected to the

Aylo Exhibit 1002
Page 222 of 1405

system bus 121 via an interface, such as a video interface 190. In addition to the monitor, computers may also include other peripheral output devices such as speakers 197 and printer 196, which may be connected through an output peripheral interface 195. A camera 192 (such as a digital/electronic still or video camera, or film/photographic scanner) capable of capturing a sequence of images 193 can also be included as an input device to the personal computer 110. Further, while just one camera is depicted, multiple cameras could be included as input devices to the personal computer 110. The images 193 from the one or more cameras are input into the computer 110 via an appropriate camera interface 194. This interface 194 is connected to the system bus 121, thereby allowing the images to be routed to and stored in the RAM 132, or one of the other data storage devices associated with the computer 110. However, it is noted that image data can be input into the computer 110 from any of the aforementioned computer-readable media as well, without requiring the use of the camera 192.

[0040] The computer 110 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 180. The remote computer 180 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to the computer 110, although only a memory storage device 181 has been illustrated in Fig. 1. The logical connections depicted in Fig. 1 include a local area network (LAN) 171 and a wide area network (WAN) 173, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0041] When used in a LAN networking environment, the computer 110 is connected to the LAN 171 through a network interface or adapter 170. When used in a WAN networking environment, the computer 110 typically includes a modem 172 or other means for establishing communications over the WAN 173, such as the Internet. The modem 172, which may be internal or external, may be connected to the system bus 121 via the user input interface 160, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 110, or portions thereof, may be stored in the remote memory storage device. By way of example, and not limitation, Fig. 1 illustrates remote application programs 185 as residing on memory device 181. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0042] The exemplary operating environment having now been discussed, the remaining parts of this description section will be devoted to a description of the program modules embodying the invention.

II. Problem Formulation

A. Temporal Coordinate Systems

[0043] It will pay to distinguish between the temporal coordinate systems, or clocks, used to express time. Herein, *media time* refers to the clock running on the device used to capture and timestamp the original content, while *client time* refers to the clock running on the client used to play back the content. It is assumed that media time is real time (i.e., one second of media time elapses in one second of real time) at the time of media capture, while client time is real time at the time of media playback. The symbol τ is used to express media time and the symbol t to express client time, with subscripts and other arguments to indicate corresponding events. For example, $\tau_d(0),\tau_d(1),\tau_d(2),...$ is used to express the playback deadlines of frames 0,1,2,... in media time, while $t_d(0),t_d(1),t_d(2),..$ is used to express the playback deadlines of frames 0,1,2,... at the client. Content may be played back at a rate v times real time. Thus the conversion from media time to client time can be expressed as

$$t = t_0 + \frac{\tau - \tau_0}{v}, \qquad (1)$$

where t_0 and τ_0 represent the time of a common initial event, such as the playback of frame 0 (or the playback of the first frame after a seek or rebuffering event) in media and client coordinate systems, respectively.

B. Leaky Bucket Model

[0044] For the moment imagine a scenario in which both the encoder 200 and the decoder 202 run in real time over an isochronous communication channel of a network 204. In this case, to match the instantaneous coding bit rate to the instantaneous channel rate, an *encoder buffer* 206 is required between the encoder 200 and the network 204 and a *decoder buffer* 208 is required between the network 204 and the decoder 202 , as illustrated in Fig. 2. A *schedule* is the sequence of times at which successive bits in the coded bit stream pass a given point in the communication pipeline.

Aylo Exhibit 1002
Page 223 of 1405

The graph in Fig. 3 illustrates the schedules of bits passing the points A, B, C, and D in Fig. 2. Schedule A is the schedule at which captured frames are instantaneously encoded and put into the encoder buffer. This schedule is a staircase in which the nth step rises by $b(n)$ bits at time $\tau(n)$, where $\tau(n)$ is the time at which frame n is encoded, and $b(n)$ is the number of bits in the resulting encoding. Schedules B and C are the schedules at which bits respectively enter and leave the communication channel. The slope of these schedules is R bits per second, where R is the communication rate of the channel. Schedule D is the schedule at which frames are removed from the decoder buffer and instantaneously decoded for presentation. Note that Schedule D is simply a shift of Schedule A. Note also that Schedule B is a lower bound to Schedule A, while Schedule C is an upper bound to Schedule D. Indeed, the gap between Schedules A and B represents, at any point in time, the size in bits of the encoder buffer, while the gap between Schedules C and D likewise represents the size of the decoder buffer. The encoder and decoder buffer sizes are complementary. Thus the coding schedule (either A or D) can be contained within a *buffer tube,* as illustrated in the graph of Fig. 4, having slope R , height B , and initial offset F^d from the top of the tube (or equivalently initial offset $F^e = B - F^d$ from the bottom of the tube). It can be seen that $D = F^d/R$ is the *startup delay* between the time that the first bit arrives at the receiver and the first frame is decoded. Thus it is of interest to minimize F^d for a given R.

[0045] A *leaky bucket* is a metaphor for the encoder buffer. The encoder dumps $b(n)$ bits into the leaky bucket at time $\tau(n)$, and the bits leak out at rate R. In general it is possible for the leak rate R to be high enough so that the bucket occasionally empties. Thus the encoder buffer fullness $F^e(n)$ immediately before frame n is added to the bucket and the encoder buffer fullness $B^e(n)$ immediately after frame n is added to the bucket evolve from an initial encoder buffer fullness $F^e(0) = F^e$ according to the dynamical system

$$B^e(n) = F^e(n) + b(n), \tag{2}$$

$$F^e(n+1) = \max\{0, B^e(n) - R/f(n)\}, \tag{3}$$

where

$$f(n) = \frac{1}{\tau(n+1) - \tau(n)} \tag{4}$$

is the instantaneous frame rate, for n=0,1,2,... If R is sufficiently low, then the bucket will never run dry (underflow), but if R is too low the bucket will eventually overflow. The largest R such that the buffer will never run dry is taken to be the average coding bit rate r_c of the bit stream. This is made more precise in the following two paragraphs.

[0046] A leaky bucket with size B, rate R, and initial fullness F^e is said to *contain* a stream having a schedule characterized by the steps $\{(b(n),\tau(n))\}$ if $B^e(n) \le B$ for all n. The minimum bucket size needed to contain the stream given leak rate R and initial fullness F^e is defined as

$$B^e_{\min}(R, F^e) = \min_n B^e(n), \tag{5}$$

while the corresponding initial decoder buffer fullness is defined as

$$F^d_{\min}(R, F^e) = B^e_{\min}(R, F^e) - F^e. \tag{6}$$

The minimum of each of these over F^e is denoted as

Aylo Exhibit 1002
Page 224 of 1405

$$B^e_{\min}(R) = \min_{F^e} B^e_{\min}(R, F^e), \tag{7}$$

$$F^d_{\min}(R) = \min_{F^e} F^d_{\min}(R, F^e). \tag{8}$$

It is shown in [10, Proposition 2] that remarkably; these are each minimized by the same value of F^e, which is hence equal to

$$F^e_{\min}(R) = B^e_{\min}(R) - F^d_{\min}(R). \tag{9}$$

Thus the bit stream with schedule $\{(b(n),\tau(n))\}$, for each bit rate R there is a unique leaky bucket that contains the stream and that has the minimum buffer size B as well as the minimum startup delay $D = F^d/R$. These parameters can be computed with the above equations.

[0047] For sufficiently low leak rates R, the leaky bucket does not underflow, when beginning with initial fullness $F^e = F^e_{\min}(R)$. Tthe maximum such rate R may be used as the average coding bit rate r_c of a bit stream with coding schedule $\{(b(n),\tau(n))\}$.

[0048] Leak rates R greater than r_c can also be employed. It is shown in [10] that both $B^e_{\min}(R)$ and $F^d_{\min}(R)$ are decreasing, piecewise linear, and convex in R. Hence if the transmission rate R is greater than the average coding bit rate r_c, the startup delay $D = F^d_{\min}(R)/R$ can be reduced compared to $D = F^d_{\min}(r_c)/R$. This fact will be used in Section IV-A.

[0049] A leaky bucket with leak rate $R=r_c$, size $B = B^e_{\min}(r_c)$ and initial decoder buffer fullness $F^d = F^d_{\min}(r_c)$ thus corresponds to a straight buffer tube bounding the coding schedule as in Fig. 4. Each stream in the media file has a coding schedule; thus each stream corresponds to a straight buffer tube with slope equal to the average coding bit rate r_c of the stream. The size B of the buffer tube and its offset F^e (or F^d) relative to the coding schedule can be either computed by the above formula for a variable bit rate (VBR) stream (such as a constant-quality sub-stream of a scalable stream), or obtained from the size B and initial state F^e of the actual encoder buffer used to encode the stream if it is a constant bit rate (CBR) stream.

[0050] In the following the gap $g(n)$ at frame n between the buffer tube *upper bound* and the coding schedule is considered, as depicted in Fig. 4. Note that the decoder buffer fullness $F^d(n) = B - F^e(n)$ can also be expressed

$$F^d(n) = b(n) + g(n) = g(n-1) + \frac{r_c(n)}{f(n)}, \tag{10}$$

where $r_c(n)$ is the coding bit rate of the buffer tube, now taking into account that different frames may lie in different buffer tubes with different coding bit rates as coding bit rate control is applied and streams are switched.

C. Rate Control Model

[0051] Assume for the moment that bits arrive at the client at a constant rate r_a. Then frame n (having size $b(n)$) arrives at the client $b(n)/r_a$ seconds after frame n-1. Indeed, the index of a bit is proportional to its arrival time. Dividing the vertical scale of the schedules in Fig. 4 by r_a, the schedules are obtained in terms of client time, rather than bits, as shown in the graph of Fig. 5. The coding schedule divided by r_a becomes the *arrival schedule,* which provides for each n the time $t_a(n)$ of arrival of frame n at the client. The buffer tube upper bound (in bits) divided by r_a becomes the buffer tube upper bound (in time), which provides for each n the time $t_b(n)$ by which frame n is guaranteed to arrive. In the

Aylo Exhibit 1002
Page 225 of 1405

same plot the *playback deadline* is shown, which is the time $t_d(n)$ at which frame n is scheduled to be played (after instantaneous decoding). Thus the gap between a frame's arrival time and its playback deadline is the client buffer duration at the time of the frame arrival. This must be non-negative to allow continuous playback.

[0052] In reality the arrival rate is not constant. If $t_a(n\text{-}1)$ and $t_a(n)$ are the arrival times of frames n and n-1 respectively, then

$$r_a(n) = \frac{b(n)}{t_a(n) - t_a(n-1)} \tag{11}$$

can be defined as the *instantaneous arrival rate* at frame n. In practice the average arrival rate at frame n is estimated by a moving average $\tilde{r}_a(n)$ of previous values of $r_a(n)$, as detailed in Section IV-C. Hence using Eq. (11) the arrival time of frame n can be expressed in terms of the arrival time of frame n-1 as

$$t_a(n) = t_a(n-1) + \frac{b(n)}{r_a(n)} \tag{12}$$

$$= t_a(n-1) + \frac{b(n)}{\tilde{r}_a(n)} + v(n), \tag{13}$$

where the $v(n)$ term is an error term that captures the effect of using the slowly moving average $\tilde{r}_a(n)$. From Eq. (10), however, it can be seen that

$$b(n) = \frac{r_c(n)}{f(n)} + g(n-1) - g(n), \tag{14}$$

whence (substituting Eq. (14) into Eq. (13)) yields

$$t_a(n) = t_a(n-1) + \frac{r_c(n)}{f(n)\tilde{r}_a(n)} + \frac{g(n-1)}{\tilde{r}_a(n)} - \frac{g(n)}{\tilde{r}_a(n)} + v(n). \tag{15}$$

Now defining the buffer tube upper bound (in time) of frame n as

$$t_b(n) = t_a(n) + \frac{g(n)}{\tilde{r}_a(n)}, \tag{16}$$

so that

Aylo Exhibit 1002
Page 226 of 1405

$$t_b(n) - t_b(n-1) = t_a(n) - t_a(n-1) + \frac{g(n)}{\tilde{r}_a(n)} - \frac{g(n-1)}{\tilde{r}_a(n-1)}, \qquad (17)$$

the following update equation is obtained:

$$t_b(n) = t_b(n-1) + \frac{r_c(n)}{f(n)\tilde{r}_a(n)} + w(n-1), \qquad (18)$$

where

$$w(n-1) = \frac{g(n-1)}{\tilde{r}_a(n)} - \frac{g(n-1)}{\tilde{r}_a(n-1)} + v(n). \qquad (19)$$

is again an error term that captures variations around a locally constant arrival rate.

[0053] Using Eq. (16), the client can compute $t_b(n\text{-}1)$ from the measured arrival time $t_a(n - 1)$, the estimated arrival rate $\tilde{r}_a(n - 1)$, and $g(n - 1)$ (which can be transmitted to the client along with the data in frame n-1 or computed at the client as described in Section V-E). Then using Eq. (18), the client can control the coding bit rate $r_c(n)$ so that $t_b(n)$ reaches a desired value, assuming the frame rate and arrival rate remain roughly constant. From this perspective, Eq. (18) can be regarded as the state transition equation of a feedback control system and it is thus possible to use a control-theoretic approach to regulate the coding bit rate.

D. Control Objective

[0054] With the state transition equation defined in Eq. (18), uninterrupted playback can be achieved by regulating the coding bit rate so that the client buffer does not underflow. To introduce a margin of safety that increases over time, a *target schedule* is introduced, as illustrated in Fig. 5, whose distance from the playback deadline grows slowly over time. By regulating the coding bit rate, it is attempted to control the buffer tube upper bound so that it tracks the target schedule. If the buffer tube upper bound is close to the target schedule, then the arrival times of all frames will certainly be earlier than their playback deadlines and thus uninterrupted playback will be ensured. Note that controlling the actual arrival times (rather than their upper bounds) to the target would result in an approximately constant number of bits per frame, which would in turn result in very poor quality overall. By taking the leaky bucket model into account, it is possible to establish a control that allows the instantaneous coding bit rate to fluctuate naturally according to the encoding complexity of the content, within previously established bounds for a given average coding bit rate.

[0055] Although controlling the upper bound to the target schedule is our primary goal, it is also desirable to minimize quality variations due to large or frequent changes to the coding bit rate. This can be achieved by introducing into the cost function a penalty for relative coding bit rate differences.

[0056] Letting $t_T(n)$ denote the target for frame n, the following cost function is used to reflect both of concerns:

$$I = \sum_{n=0}^{N} \left((t_b(n) - t_T(n))^2 + \sigma \left(\frac{r_c(n+1) - r_c(n)}{\tilde{r}_a(n)} \right)^2 \right), \qquad (20)$$

where the first term penalizes the deviation of the buffer tube upper bound from the target schedule and the second term penalizes the relative coding bit rate difference between successive frames. N is the control window size and σ is a Lagrange multiplier or weighting parameter to balance the two terms.

Aylo Exhibit 1002
Page 227 of 1405

III. Optimal Control Solution

[0057] Before presenting the optimal control solution, the design rational of the target schedule will be described.

A. Target Schedule Design

[0058] Fig. 6 is a graph showing an illustrative target schedule. The gap between the playback deadline and the target schedule is the desired client buffer duration (in client time). If the gap is small at the beginning of streaming, then it allows a small startup delay, while if the gap grows slowly over time, it gradually increases the receiver's ability to counter jitter, delays, and throughput changes.

[0059] The slope of the target schedule relates the average coding bit rate to the average arrival rate. Let $t_T(n)$ be the target for frame n. As illustrated in Fig. 6, the slope of the target schedule at frame n is

$$s(n) = \frac{t_T(n+1) - t_T(n)}{\tau(n+1) - \tau(n)}. \qquad (21)$$

If the upper bound $t_b(n)$ aligns perfectly with the target schedule (i.e., $t_b(n) = t_T(n)$) and the arrival rate r_a is constant (i.e., the $w(n\text{-}1)$ term vanishes), from Eq. (18),

$$s(n) = \frac{t_b(n+1) - t_b(n)}{\tau(n+1) - \tau(n)} = \frac{r_c(n)}{r_a}. \qquad (22)$$

Thus initially, when the slope was low, i.e., less than $1/v$, r_a/r_c is greater than v and more than v seconds of content are received per second of client time, causing the client buffer (which is playing out only v seconds of content per second of client time) to grow. Over time, as the slope approaches $1/v$, r_a/r_c approaches v and the buffer remains relatively constant (except for changes due to variations in the instantaneous coding bit rate), since content is received and played back at the same speed v. Next two target schedule functions are presented that illustrate the general design idea.

1) Logarithmic Target Schedule

[0060] One way to choose the target schedule t_T is to have the client buffer duration grow logarithmically over time. Specifically, if t_d is the playback deadline, then for each t_d greater than some start time t_{d0},

$$t_T = t_d - \frac{b}{a}\ln(a(t_d - t_{d0}) + 1). \qquad (23)$$

Since by Eq. (1), $t_d = t_{d0} + (\tau_d - \tau_{d0})/v$ this yields

$$s = \frac{dt_T}{d\tau_d} = \frac{dt_T dt_d}{dt_d d\tau_d} = \frac{1}{v} - \frac{b}{a(\tau_d - \tau_{d0}) + v}, \qquad (24)$$

and hence the initial slope at frame 0 (when $t_d = t_{d0}$) is $s(0) = (1\text{-}b)/v$. Setting b=0.5 implies that initially r_c/r_a=0.5/v, causing the client buffer to grow initially at two times real time. Further setting a=0.15 implies that the client buffer duration will be 7.68 seconds after 1 minute, 15.04 seconds after 10 minutes, and 22.68 seconds after 100 minutes, regardless of v.

Aylo Exhibit 1002
Page 228 of 1405

2) Two-piece Linear Target Schedule

[0061] Another way to choose the target schedule t_T is to have the client buffer duration grow linearly at rate b seconds of media time per second of client time until the buffer duration reaches a seconds of media time, after which it remains constant. Specifically, for each t_d greater than some start time t_{d0},

$$t_T = \begin{cases} t_d - b(t_d - t_{d0}) & t_d \le t_{d0} + a/b \\ t_d - a & t_d \ge t_{d0} + a/b \end{cases} \quad (25)$$

The initial slope is again $s(0)=(1-b)/v$. Setting b=0.5 implies that initially r_c/r_a = 05/v , causing the client buffer to grow initially at two times real time. Further setting a = 10 implies that the client buffer duration will reach 10 seconds of media time after 20 seconds of client time, regardless of v.

[0062] Figs. 7(a) and (b) show graphs of the above two target schedules. As one can see, if a client buffer duration of 10 seconds is considered to be a safe level against jitter, delay and network fluctuations, then the two-piece linear target schedule reaches the safe level in 20 seconds, much faster than the logarithmic target schedule. On the other hand, the slope of the two-piece linear target schedule remains lower for longer (hence the coding bit rate and quality are lower for longer) and furthermore experiences an abrupt change at 20 seconds when its slope changes from 0.5/v to 1/v. Consequently, the coding bit rate will not change as smoothly as with the logarithmic target schedule, although it will not be as abrupt as the schedule itself because of the smoothness objective in the controller design.

B. Optimal Controller Design

[0063] Recall from Eq. (18) the fundamental state transition equation, which describes the evolution of the buffer tube upper bound $t_b(n)$ in terms of the coding bit rate $r_c(n)$:

$$t_b(n+1) = t_b(n) + \frac{r_c(n+1)}{f\tilde{r}_a} + w(n). \quad (26)$$

Here it is now assumed that the frame rate f and the average arrival rate $\tilde{r}_a$ are relatively constant. Deviations from this assumption are captured by $w(n)$.

[0064] It is desired to control the upper bound by adjusting the coding bit rate. As each frame arrives at the client, a feedback loop can send a message to the server to adjust the coding bit rate. Note, however, that by the time frame n arrives completely at the client, frame $n+1$ has already started streaming from the server. Thus the coding bit rate $r_c(n+1)$ for frame $n+1$ must already be determined by time $t_a(n)$. Indeed, at time $t_a(n)$, frame n+2 is the earliest frame for which the controller can determine the coding bit rate. Hence at time $t_a(n)$, the controller's job must be to choose $r_c(n+2)$. This one-frame delay in the feedback loop must be explicitly accounted for.

[0065] For simplicity, the target schedule is linearized around the time that frame n arrives. The linearization is equivalent to using a line tangent to the original target schedule at a particular point as an approximate target schedule. Thus,

$$t_T(n+1) - 2t_T(n) + t_T(n-1) = 0. \quad (27)$$

Rather than directly control the evolution of the upper bound, which grows without bound, for the purposes of stability an error space formulation is used. Defining the error as

$$e(n) = t_b(n) - t_T(n), \quad (28)$$

Aylo Exhibit 1002
Page 229 of 1405

results in

$$e(n+1)-e(n) = (t_b(n+1)-t_T(n+1))-(t_b(n)-t_T(n)) \quad (29)$$

$$= (t_b(n+1)-t_b(n))-(t_T(n+1)-t_T(n)) \quad (30)$$

$$= \frac{r_c(n+1)}{f\tilde{r}_a}-(t_T(n+1)-t_T(n))+w(n), \quad (31)$$

from which in turn

$$\begin{aligned}(e(n+1)-e(n))-(e(n)-e(n-1)) &= [r_c(n+1)-r_c(n)]/f\tilde{r}_a \\ &-(t_T(n+1)-2t_T(n)+t_T(n-1)) \\ &+(w(n)-w(n-1))\end{aligned} \quad (32)$$

$$= \frac{r_c(n+1)-r_c(n)}{f\tilde{r}_a}+(w(n)-w(n-1)) \quad (33)$$

The control input is then defined as

$$u(n) = \frac{r_c(n+2)-\hat{r}_c(n+1)}{\tilde{r}_a}, \quad (34)$$

where $\hat{r}_c(n+1)$ is a possibly quantized version of $r_c(n+1)$ (as defined in Section IV-D) and the disturbance is defined as

$$d(n) = \frac{\hat{r}_c(n)-r_c(n)}{f\tilde{r}_a}+w(n)-w(n-1). \quad (35)$$

Then Eq. (33) can be rewritten

$$e(n+1) = 2e(n)-e(n-1)+\frac{u(n-1)}{f}+d(n). \quad (36)$$

Therefore , defining the error vector

Aylo Exhibit 1002
Page 230 of 1405

$$\mathbf{e}(n) = \begin{bmatrix} e(n) \\ e(n-1) \\ u(n-1) \end{bmatrix} = \begin{bmatrix} t_b(n) \\ t_b(n-1) \\ \dfrac{r_c(n+1)}{\tilde{r}_a} \end{bmatrix} - \begin{bmatrix} t_T(n) \\ t_T(n-1) \\ \dfrac{\hat{r}_c(n)}{\tilde{r}_a} \end{bmatrix} \qquad (37)$$

the error space representation of the system can be expressed as

$$\mathbf{e}(n+1) = \begin{bmatrix} 2 & -1 & \dfrac{1}{f} \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{e}(n) + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} u(n) + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} d(n), \qquad (38)$$

or $e(n+1) = \Phi e(n) + \Gamma u(n) + \Gamma_d d(n)$ for appropriate matrices Φ, Γ and Γ_d.

[0066] Assuming the disturbance *d(n)* is a pure white noise, and assuming *perfect state measurement* (i.e., all components of *e*(*n*) can be measured without using an estimator), the disturbance *d*(*n*) does *not* affect the controller design. Thus a linear controller represented by

$$u(n) = -G\mathbf{e}(n) \qquad (39)$$

can be used where G is a *feedback gain.* By the time frame *n* is completely received, all elements of **e**(*n*) are available at the client and *u*(*n*) can thus be computed. The ideal coding bit rate for frame *n* + 2 can then be computed as

$$r_c(n+2) = \hat{r}_c(n+1) - G\mathbf{e}(n)\tilde{r}_a. \qquad (40)$$

Finding the optimal linear controller amounts to finding the feedback gain *G** that minimizes the quadratic cost function defined in Section II-D. Before continuing with the design, the system *controllability matrix C*,

$$C = \begin{bmatrix} \Gamma & \Phi\Gamma & \Phi^2\Gamma \end{bmatrix} = \begin{bmatrix} 0 & \dfrac{1}{f} & \dfrac{2}{f} \\ 0 & 0 & \dfrac{1}{f} \\ 1 & 0 & 0 \end{bmatrix} \qquad (41)$$

will be checked first which has full rank for any frame rate *f*. Thus, the system is *completely controllable* and the state e(*n*) can be regulated to any desirable value. Now recall that the cost function defined in Section II-D is

Aylo Exhibit 1002
Page 231 of 1405

$$I = \sum_{n=0}^{N} \left\{ (t_b(n) - t_T(n))^2 + \sigma \left(\frac{r_c(n+1) - r_c(n)}{\tilde{r}_a} \right)^2 \right\} \quad (42)$$

$$= \sum_{n=0}^{N} \{ e(n)^T Q e(n) + u(n-1)^T R u(n-1) \}, \quad (43)$$

where $Q = C^T C$ (with C = [1 0 0]) and R = σ. Then, the original control problem of tracking the target schedule while smoothing the coding bit rate fluctuations (i.e., minimizing the cost function *I*) is converted to a standard regulator problem in the error space. Letting $N \to \infty$, the infinite horizon optimal control problem can be solved by applying the results in [11, Section 3.3] to obtain an optimal regulator in two steps: 1) solving, to get S, the *discrete algebraic Riccati equation* (DARE)

$$S = \Phi^T \{ S - S\Gamma [\Gamma^T S \Gamma + R]^{-1} \Gamma S \} \Phi + Q, \quad (44)$$

and 2) computing the optimal feedback gain

$$G^* = [\Gamma^T S \Gamma + R]^{-1} \Gamma^T S \Phi. \quad (45)$$

The existence and uniqueness of S (and in turn of G^*) is guaranteed when *Q* is nonnegative definite and R is positive definite, which is straightforward to verify in this case.

C. Frame Rate

[0067] In the previous section, it was assumed that the frame rate is constant. This assumption is reasonable when streaming a single medium, such as video without audio. Variable frame rate video is usually achieved by skipping frames, which can be accommodated by setting $b(n) = 0$. However, usually video and audio are streamed together, and their merged coding schedule may have no fixed frame rate. Even if there is a fixed frame rate *f*, it may be desirable to operate the controller at a rate lower than *f*, to reduce the feedback rate, for example.

[0068] To address these issues, in practice the notion of a *virtual frame rate* is used. A virtual frame rate *f* is chosen, for example *f* =1 frame per second (fps); media time is partitioned into intervals of size 1/*f*; and all of the (audio and video) frames arriving within each interval are modeled as a *virtual frame* whose decoding and playback deadline is the end of the interval.

[0069] This approach has several advantages. First, it allows offline design of a universal feedback gain, which is independent of the actual frame rate of the stream or streams. Second, it allows the rate of feedback from the client to the server to be reduced. And finally, since the interval between virtual frames is typically safely larger than a round trip time (RTT), a one-frame delay in the error space model (as described in the previous section) is sufficient to model the feedback delay. Otherwise it would be necessary to model the feedback delay with approximately *RTT/f* additional state variables to represent the network delay using a shift register of length *RTT/f*.

[0070] In the remainder of this description a virtual frame rate *f* = 1 fps is employed, and is referred to simply as the frame rate. Likewise, a virtual frame is referred to simply as a frame.

[0071] It is noted that with the present controller design rationale, the deviation of the buffer tube upper bound from the control target can be reduced by decreasing the σ value. A smaller value of σ value implies a relative larger penalty on the deviation term in the cost function and thus forces the upper bound to track the target more closely. This, however, happens at the cost of sacrificing coding bit rate smoothness, since the corresponding term in the cost function will be weighted less. It has been found with σ = 500 that while the buffer tube upper bound deviates only slightly from the control target, the coding bit rate has undesirable oscillations. On the other hand, a large σ value will certainly yield smoother coding bit rates, but might also incur client buffer underflow since the buffer tube upper bound is allowed to

Aylo Exhibit 1002
Page 232 of 1405

deviate significantly away from the control target. Therefore, a good choice of σ should take into account this trade-off. In tested embodiments of the present system and process, $\sigma = 4000$ was chosen when the coding bit rate switches up and $\sigma = 2000$ when it switches down. Note that a slightly more aggressive strategy is allowed in the latter case to further reduce the chance of client buffer underflow.

IV. Practical Issues with Streaming

A. Fast Startup

[0072] As discussed in previous sections, the startup delay is the length of the period from the time that content first begins to arrive at the client to the time that playback begins. During this period, content accumulates in the receiver buffer to counter packet jitter, retransmission delay, variations in network bandwidth, and variations in instantaneous coding bit rate. It is conceivable that a longer startup delay would increase the chances of being able to maintain continuous playback in a dynamic network environment. On the other hand, users expect the startup delay to be as small as possible. Thus, it is desirable to investigate techniques that can reduce the startup delay while retaining robustness. One possible approach is to transmit the content at a faster than normal rate at the beginning of streaming. This *bursting* technique will certainly build up the buffer duration in a small amount of time. It, however, puts extra pressure on the network by demanding a higher than normal initial bandwidth, which may not even be available.

[0073] The present system and process employs an alternative *fast startup* technique, which takes advantage of the properties of adaptive media. As discussed in previous sections, by choosing an initial coding bit rate r_c equal to half the arrival rate r_a (divided if necessary by the playback speed v), the client buffer duration can grow at two times real time during playback. Growing the client buffer during playback enables the startup delay to be low, because playback can begin while the buffer duration is still low. Beginning playback while the buffer duration is low is not particularly risky over the short term, because the probability of deep congestion occurring in any short interval is low. However, the probability of deep congestion occurring in a long interval is high, so it is important for the buffer duration to be high over the long term. Without the ability to grow the buffer duration during playback, startup would have to be delayed until the buffer duration was sufficiently high to guarantee continuous playback over the long term.

[0074] Moreover, if the transmission rate is twice the coding bit rate, the startup delay can be further reduced by taking advantage of properties of the leaky bucket model [10]. As detailed in Section II-B, the startup delay for a given bit stream is $D = F^d_{\min}(R)/R$ when the stream is transmitted at rate R. This is ordinarily equal to $F^d_{\min}(r_c)/r_c$ when transmitting the stream at its coding bit rate. However, when transmitting the stream at a rate $r_a > r_c$ ($r_c = 0.5r_a/v$), then the startup delay drops to $F^d_{\min}(r_a)/r_a$. Thus the startup delay D decreases both because the numerator decreases and because the denominator increases.

[0075] The graph in Fig. 8 illustrates the decrease in the initial decoder buffer fullness $F^d_{\min}(R)$ as R changes from r_c to r_a. In particular, it depicts the coding schedule for a given bit stream, as well as upper and lower bounds, denoted Tube I and Tube II, corresponding to two leaky buckets with leak rates r_c and r_a respectively, both containing the coding schedule. Tube II is smaller than Tube I, since the minimum size $B_{\min}(R)$ of a leaky bucket containing a given stream is decreasing in the leak rate R [10]. Likewise, the initial decoder buffer fullness $F_{\min}(R)$ is decreasing in R [10]. Hence the playback deadline for frame 0 can begin as early as client time $t_{0\_II} = F^d_{\min}(r_a)/r_a$, instead of $t_{0\_I} = F^d_{\min}(r_c)/r_a$. From there, the playback deadline advances at $1/v$ seconds of client time per second of media time.

B. Controller Initialization

[0076] As illustrated in Fig. 8, the target schedule starts at the same time as the playback deadline and grows according to a predefined function. The controller attempts to control the upper bound of Tube I to the target schedule. Initially the upper bound of Tube I is above the target schedule (and is indeed above the playback deadline, though it is known that this is safe). Hence, when the playback starts, the controller would try to close the gap by decreasing the coding bit rate. This, however, would not be desirable because the current coding bit rate is already lower than the arrival rate to allow the client buffer to grow. Further reduction of the coding bit rate would not be proper. To avoid this effect, the controller is initialized when the upper bound of Tube I exceeds the target schedule i.e., at point B in Fig. 8. Point B can be found analytically, but in practice there is no need to explicitly solve for it. The controller can be initialized as soon as the upper bound of Tube I exceeds the target.

Aylo Exhibit 1002
Page 233 of 1405

C. Exponential Averaging of the Arrival Rate

[0077] Using the average arrival rate (instead of the instantaneous arrival rate) helps to reduce coding bit rate oscillations. This section details a new exponential averaging algorithm for the arrival rate.

[0078] Let $\tilde{r}_a(k)$ and $r(k)$ be the average arrival rate and the instantaneous arrival rate, respectively, when packet k is received. Note that unlike the controlling operation, the rate averaging operation may be performed after the arrival of every *packet,* rather than after the arrival of every *frame.* Hence the discrete packet index k is used rather than the frame index n . Instead of using the widely adopted exponentially weighted moving average (EWMA)

$$\tilde{r}_a(k) = \beta(k)\tilde{r}_a(k-1) + (1-\beta(k))r_a(k) \qquad (46)$$

with constant $\beta(k) = \beta$, the exponential averaging is performed more carefully. In the present algorithm, the factor $\beta(k)$ is not constant, but varies according to the packets' inter-arrival gaps. This new algorithm has several advantages over the EWMA algorithm with constant $\beta(k)$. First, the estimate of the average arrival rate $\tilde{r}_a(k)$ goes to zero naturally as the gap since the last packet goes to infinity, rather than being bounded below by $\beta\tilde{r}_a(k\text{-}1)$. Second, the estimate of the average arrival rate $\tilde{r}_a(k)$ does not go to infinity as the gap since the last packet goes to zero. This is especially important, since packets often arrive in bursts, causing extremely high instantaneous arrival rates. And finally, the estimate of the average arrival rate $\tilde{r}_a(k)$ does not over-weight the initial condition, as if it represented the infinite past. This is especially important in the early stages of estimation.

[0079] As in Eq. (11), the instantaneous arrival rate after packet k is defined as

$$r_a(k) = \frac{b(k)}{t_a(k) - t_a(k-1)}, \qquad (47)$$

where here $b(k)$ denotes the size of packet k and $t_a(k)$ denotes the arrival time of packet k. The discrete time function $r_a(k)$ is extended to the piecewise constant continuous time function $r_a(t)$ by

$$r_a(t) = r_a(k) \quad \text{for all } t \in (t_a(k-1), t_a(k)), \qquad (48)$$

as illustrated in the graph in Fig. 9. Then the function $r_a(t)$ is filtered by the exponential impulse response $\alpha e^{-\alpha t}$, $t \geq 0$, for some time constant $1/\alpha$:

$$\tilde{r}_a(k) = \frac{\int_{t(0)}^{t(k)} r_a(t')\alpha e^{-\alpha(t(k)-t')}dt'}{\int_{t(0)}^{t(k)} \alpha e^{-\alpha(t(k)-t')}dt'}. \qquad (49)$$

(Here and in the remainder of this sub-section the subscript is suppressed from the arrival time $t_a(k)$.) Note that $\int_t^{\infty} \alpha e^{-\alpha t'}dt' = e^{-\alpha t}$, the denominator integral can be expressed as $1 - e^{-\alpha(t(k)-t(0))}$. Now, the range of the numerator integral is split into ranges (t(0), t(k-1)] and (t(k-1),t(k)] to obtain a recursive expression for $\tilde{r}_a(k)$ in terms of $\tilde{r}_a(k\text{-}1)$ and $r_a(k)$,

Aylo Exhibit 1002
Page 234 of 1405

$$\tilde{r}_a(k) = \frac{1-e^{-\alpha[t(k-1)-t(0)]}}{1-e^{-\alpha[t(k)-t(0)]}} e^{-\alpha[t(k)-t(k-1)]} \tilde{r}_a(k-1) + \frac{1-e^{-\alpha[t(k)-t(k-1)]}}{1-e^{-\alpha[t(k)-t(0)]}} r_a(k) \quad (50)$$

$$= \beta(k)\tilde{r}_a(k-1) + (1-\beta(k))r_a(k), \quad (51)$$

where

$$\beta(k) = \frac{e^{-\alpha[t(k)-t(k-1)]} - e^{-\alpha[t(k)-t(0)]}}{1-e^{-\alpha[t(k)-t(0)]}}. \quad (52)$$

Note that $\beta(k)$ is numerically stable as k goes to infinity. However, as the gap $\delta = t(k) - t(k\text{-}1)$ goes to zero, $1 - \beta(k)$ goes to zero while $r_a(k)$ goes to infinity. Their product, however, is well behaved. Indeed,

$$\tilde{r}_a(k) = \frac{1-e^{-\alpha[t(k-1)-t(0)]}}{1-e^{-\alpha[\delta+t(k-1)-t(0)]}} e^{-\alpha\delta} \tilde{r}_a(k-1) + \frac{1-e^{-\alpha\delta}}{1-e^{-\alpha[t(k)-t(0)]}} \frac{b(k)}{\delta} \quad (53)$$

$$\rightarrow \tilde{r}_a(k-1) + \frac{\alpha b(k)}{1-e^{-\alpha[t(k)-t(0)]}} \quad (54)$$

as $\delta \rightarrow 0$, using l'Hôpital's rule. Thus Eq. (54) is the update rule in the case when $t(k) = t(k\text{-}1)$.

D. Choosing a Stream Given a Coding bit rate

[0080] When the client requests a coding bit rate $r_c(n)$, the server complies by choosing a stream (or sub-stream of a scalable stream) having coding bit rate $\hat{r}_c(n)$ approximately equal to $r_c(n)$. There are several reasons that $\hat{r}_c(n)$ may differ from $r_c(n)$. The first reason is that there are only a finite number of streams (or sub-streams) in the media file, even if fine grain scalable coding is used. Thus there may be no stream in the media file with average coding bit rate exactly equal to $r_c(n)$. The second reason is that, even if there is a stream in the media file with average coding bit rate exactly equal to $r_c(n)$, the buffer tube for the stream may be too large to allow switching to the stream without risk of client buffer underflow. In fact, whenever the stream switches, there is generally a discontinuity in the upper bound, which may be either positive or negative. A positive shift in the upper bound is illustrated in the graph in Fig. 10, which, if large, could cause the client buffer to underflow either immediately or eventually.

[0081] Thus the server must choose a stream that causes the upper bound to shift up no more than some amount $\Delta^{max}g(n\text{-}1)$ supplied to it by the client. The client supplies $\Delta^{max}g(n\text{-}1)$ to the server in its feedback along with $r_c(n)$, shortly after client time $t_a(n\text{-}2)$ (after frame n-1 has already begun streaming). Upon receiving the feedback, the server selects a stream with coding bit rate $\hat{r}_c(n)$ as high as possible such that $\hat{r}_c(n) \le r_c(n)$ and, if $\hat{r}_c(n) > \hat{r}_c(n\text{-}1)$ (i.e., if it is a switch up in rate), then $g^{new}(n\text{-}1) - g^{old}(n\text{-}1) \le \Delta^{max}g(n\text{-}1)$, where $g^{new}(n\text{-}1)$ and $g^{old}(n\text{-}1)$ are illustrated in Fig. 10. The constraint given by $\Delta^{max}g(n\text{-}1)$ is not applied if it is a switch down in rate.

[0082] The client chooses $\Delta^{max}g(n - 1)$ to limit (its prediction of) what the upper bound would be at time $t_a(n - 1)$ if the new coding bit rate were in effect, namely,

Aylo Exhibit 1002
Page 235 of 1405

$$t_b^{new}(n-1) \approx \frac{t_b(n-2)+\hat{r}_c(n-1)}{f\tilde{r}_a}+\frac{\Delta g(n-1)}{\tilde{r}_a} \qquad (55)$$

$$\leq t_T(n-1)+p[t_d(n-1)-t_T(n-1)]. \qquad (56)$$

That is, the client chooses $\Delta^{max}g(n\text{-}1)$ to limit $t_b^{new}(n-1)$ so that it would be no more than fraction p of the way from the target $t_T(n\text{-}1)$ to the playback deadline $t_d(n\text{-}1)$. In tested embodiments of the present system and process, p=1/3 was chosen.

[0083] It is noted that in an alternate embodiment, the information as to what coding bit rates the server supports can be provided to the client ahead of time and the client can be responsible for performing the foregoing analysis. In this case, the client would only request coding bit rates that are supported by the server, and there would be no need to provide the $\Delta^{max}g(n\text{-}1)$ to the server with the request.

E. Control Target Adjustment

[0084] When a frame with a new average coding bit rate $\hat{r}_c(n)$ arrives at the client at time $t_a(n)$, there is a shift in the upper bound. This shift can be on the order of seconds and hence, rather than being negligible, can be confusing to the controller. If the shift is upward, for example, the controller will immediately try to reduce the coding bit rate $r_c(n+2)$. If the shift is downward, on the other hand, the controller will immediately try to increase the coding bit rate $r_c(n+2)$. Either way is probably not good; the intention is that $\hat{r}_c(n)$ will be maintained unless there is a disturbance in the arrival rate. One solution is to introduce a simultaneous shift in the control target schedule equal to $\Delta g(n-1)/\tilde{r}_a$; where $\Delta g(n-1) = g^{new}(n-1) - g^{old}(n\text{-}1)$ is the actual shift in the upper bound (in bits) at frame n-1 computed at the server, as illustrated in Fig. 10. The server can send this value to the client along with frame n. If there is no stream change, this value is simply zero.

[0085] If the control target schedule is adjusted whenever the coding bit rate changes, it will no longer follow the designed target schedule. The adjusted target schedule will be referred to as the *control target* schedule to distinguish it from the *designed target* schedule (or simply the *target schedule).*

[0086] The control target schedule, of course, should have a tendency to approach the designed target schedule. The basic idea is to decrease the slope of the control target schedule when it is above the designed target schedule and to increase the slope when it is below.

[0087] For the logarithmic target schedule $t_T = t_d - \frac{b}{a}\ln(at_d+1)$ (where $t_d = t_{d0} + (\tau_d - \tau_{d0})/v$), according to Eq. (24) the slope at media time τ_d is

$$s = \frac{dt_T}{d\tau_d} = \frac{1}{v} - \frac{b}{a(\tau_d-\tau_{d0})+v}. \qquad (57)$$

If d is defined as the distance between the playback deadline and the target schedule, namely

$$d = \frac{b}{a}\ln\left(a\left(\frac{\tau_d-\tau_{d0}}{v}\right)+1\right), \qquad (58)$$

then the slope may be expressed as a function of d

Aylo Exhibit 1002
Page 236 of 1405

$$s = \frac{1}{\nu} - \frac{b}{\nu e^{(a/b)d}}. \qquad (59)$$

Hence whenever d is the distance between the playback deadline and the control target, the slope of the control target is set to s in Eq. (59). Specifically, if $t_T^A(n)$ is the control target at frame n after the shift, then $t_T^A(n\text{-}1)$ is reset to be $t_T^A(n)\text{-}s/f$. Then $t_T^A(n)$ and $t_T^A(n\text{-}1)$ is used in place of $t_T(n)$ and $t_T(n\text{-}1)$ to compute the error vector $e(n)$ in Eq. (37). The resulting error vector is then used to compute the ideal coding bit rate in Eq. (40).

[0088] For the two-piece linear target schedule, the slope is easy to compute by using a predefined time period over which the control target schedule is expected to return to the target schedule. The slope of the control target schedule can then be computed from the distance d and the period. In tested embodiments of the present system and process the period was set to 50 seconds.

V. Implementation Details

[0089] This section highlights implementation details on both the sender and the receiver side.

A. Generation of Virtual Streams

[0090] In tested embodiments of the present a system and process, a fine grained scalable (FGS) stream comprises a set of data units, each tagged by a Lagrange multiplier λ representing the per-bit decrease in distortion if the data unit is received by the client. If the λ for the data unit is above a threshold, then the data unit is included in a virtual stream corresponding to that threshold. Each threshold corresponds to an overall number of bits and hence an average coding bit rate for the virtual stream. In tested embodiments, $N = 50$ virtual streams are generated. A threshold is chosen for each stream such that the resulting streams have coding bit rates that are uniformly spaced in the log domain between lower and upper bounds.

[0091] During streaming, when the server reads a data unit from the media file, it includes the data unit in the virtual stream currently being transmitted if its Lagrange multiplier λ is above the threshold for the stream.

B. Leaky Bucket Computations at the Sender

[0092] For each virtual stream, leaky bucket parameters $(R, B_{\min}(R), F_{\min}^{d}(R))$ are pre-computed offline for $R = R_{avg}$ and $R = R_{\max}$, where $R_{avg} = r_c$ is the average coding bit rate of the stream, and $R_{max} = 2r_c$. These leaky bucket parameters are sent to the client in a preamble.

[0093] In addition, during streaming the server performs on-line leaky bucket simulations for each stream. Specifically, whenever the server reads a data unit from the media file, it determines the virtual streams to which the data unit belongs, using the Lagrange multiplier of the data unit and the list of thresholds for each stream. The sender then updates, for the determined streams, the states of those leaky buckets having leak rates equal to an average coding bit rate R_{avg}, using Eqs. (2) and (3). Once all the data units in a frame are read from the media file, the sender computes $g(n) = B_{\min}(R_{avg}) - B^e(n)$ for each of the virtual streams. On a stream switch (i.e., $\hat{r}_c(n) \neq \hat{r}_c(n-1)$), the gap $g^{new}(n)$ for the new stream is transmitted to the client along with $\Delta g(n\text{-}1)=g^{new}(n\text{-}1)\text{-}g^{old}(n\text{-}1)$ as described below. It is easy to see that the cost of updating the leaky bucket states is quite low. However, it is also possible to pre-compute these values and store them with each data unit in the media file.

C. Initial Coding bit rate Selection

[0094] At the beginning of a streaming session, the sender needs to have some knowledge of the available network bandwidth so that it can choose an initial coding bit rate (usually half of the bandwidth). The bandwidth estimate can be drawn from proactive measurements, using approaches such as packet pair [12], path chirp [3], etc., or reactive approximations based on history values. The exact form of the initial bandwidth estimation is beyond the scope of this work.

D. Coding bit rate Switching

[0095] The rate control feedback from the client contains the frame number at which feedback is generated (s.g., n -

Aylo Exhibit 1002
Page 237 of 1405

2 in the previous section) and the maximum allowable shift of the upper bound in bits (e.g., $\Delta^{max}g(n\text{-}1)$ in the previous section). If the sender finds a suitable coding bit rate and makes a switch at frame n, it will transmit three values to the client along with the frame: the new coding bit rate $\hat{r}_c^{new}(n)$, the current gap to the upper bound $g^{new}(n)$, and the shift $\Delta g(n\text{-}1) = g^{new}(n\text{-}1)\text{-}g^{old}(n\text{-}1)$. With this information, the client can properly adjust its control target schedule as well as its upper bound. Note that coding bit rate switching always happens at the beginning of a new frame, never inside a frame.

E. Optimal Rate Control at the Client

[0096] Whenever a new coding bit rate starts, the client receives the value $g(n)$ along with the new frame. The values of $g(n)$ for successive frames can be then inferred by the client itself based on the coding bit rate $\hat{r}_c(n)$ and the frame size $b(n)$. The client records the arrival frame time $t_a(n)$, calculates the buffer tube upper bound $t_b(n)$ and then computes the deviation $e(n)$. If there is a coding bit rate switch, it will also compute the buffer tube shift and adjust the control target schedule accordingly. Then, $e(n)$ is fed to the optimal rate controller, which outputs a desired new coding bit rate. The latest new coding bit rate is fed back to the sender whenever there is a feedback opportunity, which could be generated at regular intervals or on-demand.

F. Exemplary Process

[0097] An exemplary process for the implementation of the foregoing system will now be provided as applicable to one embodiment of the present invention.

[0098] Referring to Figs. 11A-C, the server first computes a set of parameters defining an initial state of the encoder buffer as it would exist if a data stream corresponding to a supported coding bit rate were streamed therefrom, for each of the supported coding bit rates (process action 1100). These parameters, which include the coding bit rate associated with the data stream, the size of the encoder buffer employed with the coding bit rate of the data stream, and a value indicative of the initial encoder buffer fullness exhibited at the coding bit rate of the data stream, are provided to the client in a preamble to the streaming media data (process action 1102). The server then streams the media data to the client at a prescribed initial coding bit rate, during a startup period (process action 1104). As the data is streamed, the server computes the upper bound gap for each frame generated for each coding bit rate supported by the server (process action 1106). Initially, the client employs the encoder buffer parameters to determine on a frame-by-frame basis a new, optimal coding bit rate for a future frame (process action 1108). In addition, the client computes the aforementioned maximum allowable difference value for the frame generated by the server just prior to the future frame, i.e., the maximum shift value (process action 1110). The client also begins playback of the frames it has received as soon as the client's decoder buffer has the minimum amount of data therein that is needed to ensure an underflow condition does not occur at the initial coding bit rate and continues to playback each frame received thereafter (process action 1112). In addition, once the aforementioned startup period is over, the client requests that the streaming media data be supplied at a new coding bit rate corresponding to the determined optimal coding rate, starting at the future frame, and provides the maximum shift value along with the request (process action 1114).

[0099] The server finds a supported coding bit rate that is equal to, or if none are equal, the closest smaller rate to, the optimum coding bit rate requested by the client (process action 1116). It is next determined by the server if the supported coding bit rate found is lower or higher than the coding bit rate associated with the frame generated immediately prior to the frame specified in the client request (process action 1118). If it is lower, then server computes the aforementioned shift value for the frame generated immediately prior to the frame specified in the client request (process action 1120), and generates all future frames beginning with the frame specified by the client at the supported rate found and streams them to the client, along with the computed upper bound gap and shift value, and an indication of the new coding bit rate (process action 1122). However, if the supported coding bit rate found is higher, then server computes the shift value for the frame generated immediately prior to the frame specified in the client request (process action 1124) and then determines if the shift value is less than or equal to the maximum shift value provided in the client request (process action 1126). If the shift value is less than or equal to the maximum shift value, then the server generates all future frames beginning with the frame specified by the client at the supported coding bit rate found and streams them to the client, along with the computed upper bound gap and shift value, and an indication of the new coding bit rate (process action 1122). But, if the shift value is greater than the maximum shift value, then the next lower supported coding bit rate is found (process action 1128) and the process actions 1118 through 1128 are repeated.

[0100] The client shifts the currently scheduled target arrival time associated with a just received first frame at the new coding bit rate by the shift value provided, and shifts the currently scheduled target arrival times for future frames such that they collectively approach, over a period of time, and eventually coincide with, the previous target arrival times for those frames (process action 1130). The client then employs the encoder buffer parameters and the upper bound gap value provided by the server to determine a new, optimal coding bit rate for a future frame and to compute the

Aylo Exhibit 1002
Page 238 of 1405

maximum shift value for the frame generated by the server just prior to the future frame (process action 1132). The client requests that the streaming media data be supplied at a new coding bit rate corresponding to the determined optimal coding rate starting at the future frame specified by the client, and includes the computed maximum shift value (process action 1134). Process actions 1116 through 1134 are then repeated for the duration of the streaming media data transmission.

VI. Multiple Bit Rate Streaming

[0101] Multiple bit rate (MBR) streaming is a network adaptive technique that is widely used in commercial streaming media systems. In MBR streaming, in contrast to scalable streaming, the content is encoded into several (typically 5-7) independent streams at different coding bit rates. Often, each stream is optimized for a common type of network connection (e.g., dial-up, DSL, cable). During an MBR streaming session, the proper coding bit rate is dynamically selected based on the available network bandwidth, with the goal of achieving the maximum possible quality under the condition of uninterrupted playback. It is easy to see that MBR streaming is analogous to scalable streaming. Indeed MBR streaming can be viewed as a special case of scalable streaming with a limited number of coding bit rates available. Hence, the foregoing optimal control approach should be applicable to this case.

[0102] There are, however, several differences that complicate MBR streaming, which need to be carefully addressed. First, as just mentioned, in MBR streaming there are only a limited number of coding bit rates available. This coarse quantization of the desired coding bit rate introduces a significant nonlinearity into the closed loop system. In fact, the large gaps between the available coding bit rates introduce oscillations. For example, if two neighboring coding bit rates straddle a constant arrival rate, the controller will oscillate between the two coding bit rates in an attempt to keep the client buffer at a target level.

[0103] Second, in MBR streaming the coding bit rate cannot be switched at an arbitrary time. In fact, before the server can switch to a new stream, it may have to wait for the next clean point (e.g., I frame) in the new stream, which could be five or ten seconds away. Thus, the old coding bit rate may continue for quite a while before it changes to the new coding bit rate. From the controller's perspective, this long random extra delay tends to destabilize the closed-loop system.

[0104] Third and finally, in MBR streaming, server performance issues are critical. The commercial-grade streaming media systems that use MBR streaming do so because of the minimal computational load that it imposes on the server compared to scalable streaming. Thus, for MBR streaming it is important to keep almost all computation and state maintenance on the client side. In particular, the server will not be able to update the leaky bucket information for each stream. Instead, the client must use some mechanism for estimating and maintaining this information.

A. Conservative Up-Switching

[0105] In this subsection, a technique to help stabilize the control system and reduce steady state oscillations to a period of at least a minute is described. With this technique, rapid down-switching is permitted. In fact, the value of σ is reduced from 2000 to 500, changing the balance between responsiveness and smoothness of the coding bit rate in favor of rapid switching response. However, only conservative up-switching is permitted. Conservative up-switching ensures that spurious changes in coding bit rate do not occur, and that oscillations in the coding bit rate have a low frequency. In particular, conservative up-switching reduces the oscillations between two adjacent but widely spaced MBR coding bit rates, one above the arrival rate and one below the arrival rate.

[0106] In one embodiment, conservative up-switching establishes a conservative limit that limits the coding bit rate to no more than the arrival rate. However, in another embodiment, the method behind conservative up-switching is to establish a conservative limit on how high the coding bit rate can be raised above the arrival rate. If the current coding bit rate is below the arrival rate, and the client buffer duration begins to increase above its target level, then the coding bit rate can be switched up to a new coding bit rate above the arrival rate only if the new coding bit rate is below the conservative limit. When the client buffer duration begins at the target level, the conservative limit is equal to the arrival rate. However, as the client buffer duration increases, the conservative limit increases as well. Thus, if the current coding bit rate is below the arrival rate, and the next higher coding bit rate is above the arrival rate, then it will be possible to switch up to the next higher coding bit rate only after the client buffer duration has increased sufficiently so that the conservative limit rises above the higher coding bit rate. Once the coding bit rate is switched up to the higher coding bit rate, the client buffer begins to drain since the coding bit rate is then above the arrival rate. Eventually, when the buffer drains back below its target level, the controller will rapidly switch the coding bit rate back down to the coding bit rate below the arrival rate.

[0107] Given the current client buffer duration, the conservative limit is set to a value such that if the coding bit rate is switched up to a new coding bit rate at this value, the client buffer would take at least Δt seconds of client time to drain back to the target level. Thus, the mechanism ensures that the period of oscillation will be at least Δt seconds. In tested embodiments of the present system and process, Δt is set to 60 seconds.

Aylo Exhibit 1002
Page 239 of 1405

[0108] Fig. 12 shows how the conservative limit is computed. Let $\Delta\tau_1$ be the client buffer duration (in media time) at the moment that the coding bit rate is switched up from r_c^{old} to r_c^{new}. Thus $\Delta\tau_1$ is the number of seconds of content that will be consumed at the old coding bit rate r_c^{old} before content at the new coding bit rate begins to be consumed. (For simplicity it is assumed that all of the content in the client buffer at the time of the switch is coded at rate r_c^{old}.) Let $\Delta\tau_2$ be the number of seconds of content that is consumed at the new coding bit rate r_c^{new} before the client buffer duration drops to some level $\Delta\tau_3$ seconds (in media time), greater than the target level $\Delta\tau_T$. The duration of this phase is determined such that the total time since the switch is exactly $\Delta t = (\Delta\tau_1 + \Delta\tau_2)/v$ seconds (in client time). Now, the number of bits that arrive in this time is $r_a \Delta t = r_c^{new}(\Delta\tau_2 + \Delta\tau_3) \geq r_c^{new}(\Delta\tau_2 + \Delta\tau_T) = r_c^{new}(v\Delta t - \Delta\tau_1 + \Delta\tau_T)$, or

$$r_c^{new} \leq \frac{r_a \Delta t}{v\Delta t - \Delta\tau_1 + v\Delta t_T}, \tag{60}$$

where Δt_T is the target buffer duration in client time. The parameter Δt can be tuned to yield the desired behavior. A large Δt means that up-switching will be more conservative, while a smaller Δt means that up-switching will be more prompt. In tested embodiments, Δt is set to 60 seconds while the target Δt_T is typically about 10 seconds.

B. Buffer Tube Upper Bound Estimation

[0109] In Section V-D it was specified that the server sends three values to the client at the beginning of each change in coding bit rate: the new coding bit rate $\hat{r}_c^{new}$, the current gap to the upper bound $g^{new}(n)$, and the control target shift $\Delta g(n-1) = g^{new}(n-1) - g^{old}(n-1)$. The server computes the latter two values by running a leaky bucket simulator for each coding bit rate. The client continues to update $g(n)$ for the new coding bit rate by running its own leaky bucket simulator for the new coding bit rate. That is, beginning with the initial condition $F^e(n) = B - b(n) - g^{new}(n)$, for each successive frame the client computes

$$B^e(n) = F^e(n) + b(n) \tag{61}$$

$$F^e(n+1) = \max\{0, B^e(n) - \hat{r}_c f(n)\}, \tag{62}$$

where

$$f(n) = \frac{1}{\tau(n+1) - \tau(n)} \tag{63}$$

is the instantaneous frame rate, as in Eqs. (2), (3) and (4). From this, the client can compute

$$g(n) = B - B^e(n) \tag{64}$$

for each frame.

Aylo Exhibit 1002
Page 240 of 1405

[0110] However, if the server is unable to simulate the leaky buckets and cannot send $g^{new}(n)$ to the client, then the client must estimate this information for itself. In this case it is recommended that the client estimates $g^{new}(n)$ as an upper bound such as $\hat{g}^{new}(n)=B-b(n)\geq g^{new}(n)$. Then, beginning with initial condition $\hat{F}^{e}(n)=B-b(n)-g^{new}(n)$ (which equals 0 in this case), for each successive frame the client computes

$$\hat{B}^{e}(n) = \hat{F}^{e}(n) + b(n) \tag{65}$$

$$\hat{F}^{e}(n+1) = \max\{0, \hat{B}^{e}(n) - \hat{r}_c/f(n)\}, \tag{66}$$

as well as

$$\hat{g}(n) = B - \hat{B}^{e}(n). \tag{67}$$

It is easy to see by induction that $\hat{F}^{e}(n) \leq F^{e}(n)$, $\hat{B}^{e}(n) \leq B^{e}(n)$, and $\hat{g}(n) \geq g(n)$. Moreover, these bounds each become tighter by $\delta(n) = \hat{r}_c/f(n) - B^{e}(n)$ whenever $\delta(n) > 0$, i.e., whenever $F^{e}(n+1)$ is clipped to 0 in Eq. (66). In fact, given enough time they may eventually become tight.

[0111] Note that whenever the bounds tighten by $\delta(n) > 0$, the control target must be shifted by $\Delta g(n)/\tilde{r}_a$, where $\Delta g(n) = -\delta(n)$. Furthermore, whenever n is the first frame of a new coding bit rate, the control target should be shifted by $\Delta g(n)/\tilde{r}_a$, where $\Delta g(n) = \hat{g}^{new}(n)-\hat{g}^{old}(n)$. Here, $\hat{g}^{old}(n)$ can be determined by running Eqs. (65), (66) and (67) for one extra step, namely if n is the first frame of the new coding bit rate,

$$\hat{F}^{e}(n) = \max\{0, \hat{B}^{e}(n-1) - \hat{r}_c^{old}/f(n-1)\} \tag{68}$$

$$\hat{B}^{e}(n) = \hat{F}^{e}(n) + b(n) \tag{69}$$

$$\hat{g}^{old}(n) = B - \hat{B}^{e}(n). \tag{70}$$

It is easy to see that if $\hat{g}^{new}(n) = B - b(n)$, then $\Delta g(n)=\hat{F}^{e}(n)$ as computed in Eq. (68).

VII. Sender-Driven Streaming

[0112] The preceding discussions have assumed that the control process for adapting the stream or its bit rate to the prevailing conditions is located within the client. This is called *receiver-driven* streaming because the client (i.e., the receiver) makes the decisions and informs the server (i.e., the sender). However, it should be clear that sender-driven streaming is also possible, in which the control process operates at the sending side. For example, using TCP for transmission of the streaming media data, the control process can operate at the server, measuring the transmission times of each frame and estimating the transmission rate, instead of the control process operating at the client, measuring the arrival times of each frame and estimating the arrival rate. Using TCP, the transmission time of a frame is substantially the same as its arrival time, and likewise the transmission rate is substantially the same as its arrival rate; hence these can be used interchangeably. Generally, the transmission time or the arrival time of a packet may be called a time stamp. In sender-driven streaming, the server can infer the state of the client buffer by knowing what it has already transmitted and by knowing the playout times for the transmitted frames. Sender-driven streaming reduces the overhead of communicating control information between the client and the server, but it also increases the computational burden on the server. In some situations, this may be a desirable trade-off. Those skilled in the art will appreciate that it is possible to appropriately modify the protocols described herein to cases where the control algorithm resides in the server or in

Aylo Exhibit 1002
Page 241 of 1405

another location, rather than in the client.

VIII. References

[0113]

[1] G.J. Conklin, G.S. Greenbaum, K.O. Lillevold, A.F. Lippman, and Y.A. Reznik. Video coding for streaming media delivery on the Internet. *IEEE Trans. Circuits and Systems for Video Technology,* 11 (3):269-281, March 2001. special issue on Streaming Video.

[2] T. Wiegand and G. Sullivan. Joint video specification rec. h.264 & 14496-10 avc. Non-Final Draft of Final Draft International Standard (FDIS) JVT-G050, ITU-T & ISO/IEC, Pattaya, Thailand, March 2003.

[3] V. J. Ribeiro, R. H. Riedi, J. Navratil, L. Cottrell, and R. G. Baraniuk. pathchirp: efficient available bandwidth estimation for network paths. In *Proc. Passive and Active Measurement Workshop (PAM)*, La Jolla, CA, April 2003.

[4] R. Rejaie, M. Handley, and D. Estrin. Layered quality adap-tation for Internet streaming video. *IEEE J. Selected Areas in Communications,* 18(12):2530-2543, December 2000.

[5] Q. Zhang, Y.-Q. Zhang, and W. Zhu. Resource allocation for multimedia streaming over the Internet. *IEEE Trans. Multimedia*, 3(3):339-355, September 2001.

[6] B.-J. Kim, Z. Xiong, and W. A. Pearlman. Low bit-rate scalable video coding with 3D set partitioning in hierarchical trees (3-D SPIHT). *IEEE Trans. Circuits and Systems for Video Technology,* 10(8):1374-1387, December 2000.

[7] F. Wu, S. Li, and Y.-Q. Zhang. A framework for efficient progressive fine granularity scalable video coding. *IEEE Trans. Circuits and Systems for Video Technology,* 11(3):301-317, March 2001.

[8] J. Li. Embedded audio coding (eac) with implicit psychoacoustic masking. In *Proc. Int'l Conf. Multimedia,* pages 592-601, Nice, France, December 2002. ACM.

[9] P. de Cuetos and K. W. Ross. Adaptive rate control for streaming stored fine-grained scalable video. In *Proc. Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDA V)*, Miami Beach, FL, May 2002.

[10] J. Ribas-Corbera, P. A. Chou, and S. Regunathan. A generalized hypothetical reference decoder for H.264/AVC. *IEEE Trans. Circuits and Systems for Video Technology,* 13(7), July 2003.

[11] B. D. O. Anderson and J. B. Moore. *Optimal Control: Linear Quadratic Methods.* Prentice Hall, 1990.

[12] S. Keshav. Packet-pair flow control. http://www.cs.cornell.edu/-skeshav/papers.html.

[13] P. A. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," *Microsoft Research Technical Report MSR-TR-2001-35*, February 2001

Claims

1. A computer-implemented process for controlling a coding bit rate of streaming media data being transmitted to a client from a server over a computer network, comprising performing the following process actions:

   employing a linear quadratic control technique to establish, on an ongoing basis, a current coding bit rate for the streaming media data, which is estimated to provide a high quality playback of the streaming media data while still keeping a decoder buffer of the client used to receive streaming media data from the server filled to a desired duration level so as to reduce the chance of an underflow condition which would result in an interruption of the playback of the streaming media data by the client;
   using each coding bit rate established to control said rate for the streaming media data.

Aylo Exhibit 1002
Page 242 of 1405

2. The process of Claim 1, wherein the process action of establishing the current coding bit rate comprises the actions of:

determining on a frame-by-frame basis the coding bit rate which is estimated to keep the client buffer level approximately at the desired target level; and
establishing a new coding bit rate only when it is determined the coding bit rate estimate will not change in relation to the last established rate by more than a desired degree.

3. The process of Claim 2, wherein the client buffer level is determined by a leaky bucket model of the streaming media data at the estimated coding bit rate.

4. The process of Claim 3, wherein the process action of determining on a frame-by-frame basis the coding bit rate which is estimated will keep the client buffer approximately at the desired level, comprises an action of identifying a coding bit rate that reduces a difference between an estimated latest anticipated arrival time of the frame under consideration and a prescribed target arrival time.

5. The process of Claim 4, wherein the process action of determining on a frame-by-frame basis the coding bit rate which is estimated will keep the client buffer approximately at the desired level, comprises an action of reducing the change in the coding bit rate to a prescribed degree, using a quadratic cost function.

6. The process of Claim 5, wherein the process action of estimating the latest anticipated arrival time of the frame under consideration, comprises the actions of:

re-estimating the latest anticipated arrival time associated with the frame immediately preceding the frame under consideration if it were to exhibit the current coding bit rate; and
adding said re-estimated time to the current coding bit rate divided by the product of a current instantaneous frame rate and an estimated arrival rate of the frames.

7. The process of Claim 6, wherein the process action of re-estimating the latest anticipated arrival time associated with the frame immediately preceding the frame under consideration comprises adding the actual arrival time of said immediately preceding frame as measured by the client, to an upper bound gap of an encoder buffer of the server for the immediately preceding frame divided by the estimated arrival rate associated with said immediately preceding frame, wherein the upper bound gap of a frame is defined as the number of data bits that the server's encoder buffer can contain over the bits currently contained therein after the frame is fully input into the encoder buffer.

8. The process of Claim 7, wherein the process action of establishing the current coding bit rate is accomplished by the client, and wherein the client estimates the latest anticipated arrival time of the frame under consideration based on information received from the server concerning the state of the encoder buffer.

9. The process of Claim 8, wherein the information received from the server comprises the current coding bit rate of the streaming media data being received by the client which is received as part of the first frame of the sequence of frames exhibiting that coding bit rate.

10. The process of Claim 9, further comprising, an action of estimating the upper bound gap for each frame received by the client, said estimating comprising:

computing what the encoder buffer fullness value would have been after the last received frame was placed therein as the sum of the last computed value of the encoder buffer fullness prior to insertion of the last received frame and the size of that last received frame,

wherein the encoder buffer fullness value prior to insertion of the last received frame associated with the first frame received by the client at the current coding bit rate of the streaming media is assumed to be zero regardless of its actual value,
and wherein the encoder buffer fullness value prior to insertion of the last received frame associated with each subsequent frame received by the client at the current coding bit rate is deemed to be either zero, or the difference between the last computed value for the encoder buffer fullness after the insertion of the last received frame and the coding bit rate divided by the instantaneous frame rate, whichever is larger, wherein the instantaneous frame rate is equal to the reciprocal of a scheduled playback time of the next frame to be received less a scheduled playback time of the last received frame; and

Aylo Exhibit 1002
Page 243 of 1405

computing the upper bound gap associated with the last received frame as the encoder buffer size less the last computed value for the encoder buffer fullness after insertion of the last received frame.

11. The process of Claim 10, wherein the encoder buffer size varies depending on the coding bit rate and is received by the client from the server as a preamble to the streaming media data for all coding bit rates available from the server.

12. The process of Claim 10, further comprising, whenever a frame is received by the client which has a different coding bit rate than the immediately preceding frame, the actions of:

shifting the currently scheduled target arrival time associated with the just received frame by a shift value, wherein said shift value represents a difference between the upper bound gap of the frame and the upper bound gap that would be associated with the frame had it been coded at the immediately preceding coding bit rate;
shifting the currently scheduled target arrival times for future frames such that they, collectively approach over a period of time, and eventually coincide with, the previous target arrival times for those frames; and
shifting each shifted frame by the difference between the encoder buffer fullness value after insertion of the last received frame and the current coding bit rate, divided by the frame rate, and then divided by the estimated arrival rate associated with the last received frame.

13. The process of Claim 12, wherein the upper bound gap that would be associated with the frame had it been coded at the immediately preceding coding bit rate is computed as the encoder buffer size less the last computed value for the encoder buffer fullness value after insertion of the last received frame, wherein the encoder buffer fullness value after insertion of the last received frame is computed as the sum of the last computed value of the encoder buffer fullness value prior to insertion of the last received frame and the size of that last received frame, and wherein the encoder buffer fullness value prior to insertion of the last received frame is computed as either zero, or the difference between the value computed for the encoder buffer fullness after the insertion of the frame received prior to the last received frame and the immediately preceding coding bit rate divided by the instantaneous frame rate associated with the frame received prior to the last received frame, whichever is larger.

14. The process of Claim 8, wherein the information received from the server comprises:

the current coding bit rate of the streaming media data being received by the client which is received as part of the first frame of the sequence of frames exhibiting that coding bit rate; and
the current upper bound gap which is received as part of every frame.

15. The process of Claim 8, wherein the information received from the server comprises:

the current coding bit rate of the streaming media data being received by the client which is received as part of the first frame of the sequence of frames exhibiting that coding bit rate; and
the current upper bound gap associated with the first frame of the sequence of frames exhibiting the current coding bit rate, said current upper bound gap also being received as part of the first frame of said sequence.

16. The process of Claim 15, wherein the information received from the server further comprises a shift value which is received as part of the first frame of the sequence of frames exhibiting a coding bit rate that is different from the immediately preceding frame, said shift value representing a difference between the upper bound gap that would be associated with the immediately preceding frame received by the client had it been coded at the current coding bit rate and the upper bound gap actually associated with the immediately preceding frame coded at the previous coding bit rate.

17. The process of Claim 16, further comprising an action of whenever a frame is received by the client which has a different coding bit rate than the immediately preceding frame, shifting the currently scheduled target arrival time associated with the just received frame by the shift value and shifting the currently scheduled target arrival times for future frames such that they collectively approach over a period of time and eventually coincide with the previous target arrival times for those frames, said shifting of the target values being accomplished so as to prevent the client from attempting to establish new coding bit rates based on the shift in the upper bound gap between the old and new coding bit rates rather than a change in the arrival rate of the frames.

18. The process of Claim 15, further comprising, for each frame received by the client that is not the first frame of the sequence of frames exhibiting the current coding bit rate, an action of estimating the upper bound gap.

Aylo Exhibit 1002
Page 244 of 1405

19. The process of Claim 6, wherein computing the estimated arrival rate of the frames of the streaming media data comprises:

computing, on a packet-by-packet basis, the product of the estimated arrival rate computed for the immediately preceding packet to the currently received packet and a first fractional weighting factor, added to the product of the instantaneous arrival rate of the currently received packet and a second fractional weighting factor, wherein at least one of the fractional weighting factors is not a constant but instead based on the time between the packets.

20. The process of Claim 5, wherein the process action of identifying the coding bit rate that reduces the difference between the estimated latest anticipated arrival time of the frame under consideration and the prescribed target arrival time, while at the same time reducing the change in the coding bit rate to a prescribed degree, comprises an action of choosing the prescribed target arrival time for the frame under consideration so as to be a prescribed amount of time earlier than a scheduled playback time for that frame.

21. The process of Claim 20, wherein the process action of choosing the prescribed target arrival time for the frame under consideration comprises an action of making said prescribed amount of time large enough after a startup period that network jitter, delays and throughput changes which may cause the actual arrival time of the frame to be after its target arrival time do not result in the frame arriving after its scheduled playback time.

22. The process of Claim 21, wherein the process action of choosing the prescribed target arrival time for the frame under consideration further comprises making the target arrival time closer to the playback time during the startup period than afterwards for those frames arriving during that period to assist in reducing a startup delay between when the streaming media data begins to be received by the client and when the first frame is played back.

23. The process of Claim 22, wherein the process action of choosing the prescribed target arrival time for the frame under consideration further comprises setting the target arrival time for each frame of the streaming media using a logarithmic target schedule.

24. The process of Claim 22, wherein the process action of choosing the prescribed target arrival time for the frame under consideration further comprises setting the target arrival time for each frame of the streaming media using a two-piece linear target schedule wherein the difference between the target arrival time and the playback time for a frame arriving during the startup period increases linearly to a prescribed amount of time that is large enough to account for said network jitter, delays and throughput changes after which the difference remains substantially constant.

25. The process of Claim 22, further comprising a process action of reducing the startup delay, said reducing action comprising beginning playback of the first frame of the streaming media data as early as when a decoder buffer of the client has a minimum amount of data therein that is needed to ensure an underflow condition does not occur at an initial coding bit rate.

26. The process of Claim 25, wherein the process action of beginning playback of the first frame of the streaming media data as early as when the decoder buffer of the client has a minimum amount of data therein, comprises the actions of:

obtaining an initial encoder buffer fullness value associated with said first frame, wherein the initial encoding buffer fullness value is defined as the size of the encoder buffer less the size of the first frame and less the upper bound gap associated with the first frame;
establishing said minimum amount of data as the encoder buffer size associated with the initial coding bit rate less an initial encoder buffer fullness.

27. The process of Claim 26, wherein the initial coding bit rate, initial encoder buffer fullness, and encoder buffer size are received from the server as a preamble to the streaming media data.

28. The process of Claim 26, wherein the process action of reducing the startup delay further comprises setting the initial coding bit rate to a level that is less than the anticipated arrival rate of the streaming media data at the client since said minimum amount of data decreases as the arrival rate increases relative to the coding bit rate and the startup delay is substantially said minimum amount of data divided by the arrival rate so if the coding bit rate is less than the arrival rate, said minimum amount of data required may be smaller and the startup delay may be less.

Aylo Exhibit 1002
Page 245 of 1405

29. The process of Claim 5 wherein the process action of establishing the current coding bit rate comprises an action of initiating the establishment of coding bit rates only after a startup period, wherein said startup period is defined as the period of time prior to the first instance of an estimated latest arrival time of a frame under consideration computed for the initial coding bit rate being earlier than a target arrival time for that frame.

30. The process of Claim 5, wherein the prescribed degree to which the change in the coding bit rate is reduced varies depending on whether the coding bit rate increased, in which case the prescribed degree to which any future change in the coding bit rate is reduced is made greater in comparison to the prescribed degree in cases where the last change in the coding bit rate decreased the rate.

31. The process of Claim 3, wherein the process action of establishing a current coding bit rate for the streaming media data, comprises, in the case where the determined coding bit rate represents an increase from the immediately previous coding bit rate, establishing a new coding bit rate only if the new rate does not exceed the current estimated arrival rate of the frames.

32. The process of Claim 3, wherein the process action of establishing a current coding bit rate for the streaming media data, comprises, in the case where the determined coding bit rate represents an increase from the immediately previous coding bit rate which exceeds the current estimated arrival rate of the frames, establishing a new coding bit rate only if the current client buffer exceeds the desired level by an amount that it is estimated by the client will not be expended at the higher coding bit rate prior to the passing of a prescribed period of time.

33. The process of Claim 3, wherein the process action of establishing a current coding bit rate for the streaming media data, comprises, in the case where the determined coding bit rate represents a decrease from the immediately previous coding bit rate, establishing a new coding bit rate even if it represents a significant departure from the immediately previous coding bit rate.

34. A computer-implemented process for controlling a coding bit rate of streaming media data being transmitted to a client from a server over a computer network, comprising using the server to perform the following process actions:

generating a streaming media data stream which exhibits a coding bit rate supported by the server, and which after an initial startup period exhibits a coding bit rate related to a rate requested by the client;
computing parameters indicative of the state of an encoder buffer of the server, said encoder buffer being used to stage portions of the streaming media data prior to transmission to the client; and
providing the encoder buffer state parameters and the streaming media data stream to the client, whereby said parameters are usable by the client to establish, on an ongoing basis, a current coding bit rate for the streaming media data, which is estimated will provide a high quality playback of the streaming media data while still keeping a decoder buffer of the client used to receive streaming media data from the server filled to a desired duration level so as to reduce the chance of an underflow condition which would result in an interruption of the playback of the streaming media data by the client.

35. The process of Claim 34, wherein the process action of computing parameters indicative of the state of an encoder buffer, comprises an action of, for each coding bit rate supported by the server, computing a set of parameters defining an initial state of the encoder buffer as it would exist if a data stream corresponding to the coding bit rate where streamed therefrom, wherein said encoder buffer parameters comprise the coding bit rate associated with the data stream, the size of the encoder buffer employed with the coding bit rate of the data stream, and a value indicative of the initial encoder buffer fullness exhibited at the coding bit rate of the data stream.

36. The process of Claim 35, wherein the process action of providing the encoder buffer state parameters and the streaming media data stream to the client, comprises an action of providing the encoder buffer parameters computed for each data stream exhibiting one of the supported coding bit rates to the client as part of a preamble sent to the client prior to sending the streaming media data.

37. The process of Claim 35, wherein the size of the encoder buffer employed with the coding bit rate of the data stream is chosen so as to be the minimum size buffer that will still contain the data stream at any point in the streaming process given the coding bit rate and the initial encoder buffer fullness.

38. The process of Claim 35, further comprising a process action of the server computing an upper bound gap for each frame of the streaming media data generated for each coding bit rate supported by the server, wherein the upper

Aylo Exhibit 1002
Page 246 of 1405

bound gap is defined as the number of data bits that the server's encoder buffer can contain over the bits currently contained therein after the just-generated frame is fully input into the encoder buffer.

39. The process of Claim 38, further comprising a process action of providing the upper bound gap computed for the first frame of a sequence of frames exhibiting a coding bit rate that is different from that associated with the frame generated immediately before the first frame of said sequence of frames, to the client along with an indication of the coding bit rate associated with the sequence of frames, said gap value and coding bit rate being provided with said first frame of the sequence.

40. The process of Claim 39, wherein the client estimates the upper bound gap for frames provided by the server that do not include the associated gap value.

41. The process of Claim 40, wherein the client employs the encoder buffer parameters and the upper bound gap values to determine on a frame-by-frame basis the coding bit rate that will keep the client buffer approximately at the desired level, said determining comprising identifying a coding bit rate that reduces a difference between an estimated latest anticipated arrival time of the frame under consideration and a prescribed target arrival time.

42. The process of Claim 39, further comprising the process actions of:

the server computing a shift value for each frame representing the first frame of the sequence of frames exhibiting a coding bit rate that is different from the frame generated immediately before this first frame of the sequence, said shift value representing a difference between the upper bound gap that would be associated with the frame generated immediately before the first frame of the sequence had it been encoded at the current coding bit rate and the upper bound gap actually associated with the frame generated immediately before the first frame of the sequence encoded at the previous coding bit rate; and
providing the shift value computed for the first frame of each sequence of frames exhibiting a coding bit rate that is different from the frame generated immediately before the first frame of the sequence, to the client with the data comprising this first frame.

43. The process of Claim 42, wherein whenever a frame is provided to the client which has a different coding bit rate than the immediately preceding frame, the currently scheduled target arrival time associated with the just received frame is shifted by the shift value provided with the just-received frame and the currently scheduled target arrival times for future frames are shifted such that they collectively approach over a period of time and eventually coincide with the previous target arrival times for those frames.

44. The process of Claim 38, wherein a limited number of coding bit rates are supported by the server, and wherein the process action of generating a streaming media data stream which exhibits a coding bit rate supported by the server, and which after an initial startup period exhibits a coding bit rate related to a rate requested by the client, comprises the actions of:

(a) receiving a coding bit rate request from the client;
(b) finding a supported coding bit rate that is equal to, or if none are equal, the closest smaller rate to, the requested coding bit rate;
(c) whenever a supported coding bit rate is found that is a lower rate than the coding bit rate associated with the last generated frame of the streaming media data, generating all future frames at that supported rate; and
(d) whenever a supported coding bit rate is found that is a higher rate than the coding bit rate associated with the last generated frame of the streaming media data, determining if a difference, between the upper bound gap associated with the last generated frame had it been encoded at the supported coding bit rate found and the upper bound gap associated with that frame encoded at the current coding bit rate, is less than or equal to a maximum allowable difference value,

whenever the difference is less than the maximum allowable difference value, generating all future frames at the found supported rate, and
whenever the difference is not less than the maximum allowable difference value, finding the next lower supported coding bit rate and repeat actions (c)-(d).

45. The process of Claim 44, wherein the maximum allowable difference value is received from the client along with the coding bit rate request.

Aylo Exhibit 1002
Page 247 of 1405

46. The process of Claim 45, wherein the client chooses the maximum allowable difference value such that a latest anticipated arrival time associated with a frame provided by the server immediately prior to a frame under consideration by the client had it been coded at the requested coding bit rate is no more than a prescribed fraction of the way from that frame's target arrival time to its playback deadline.

Aylo Exhibit 1002
Page 248 of 1405

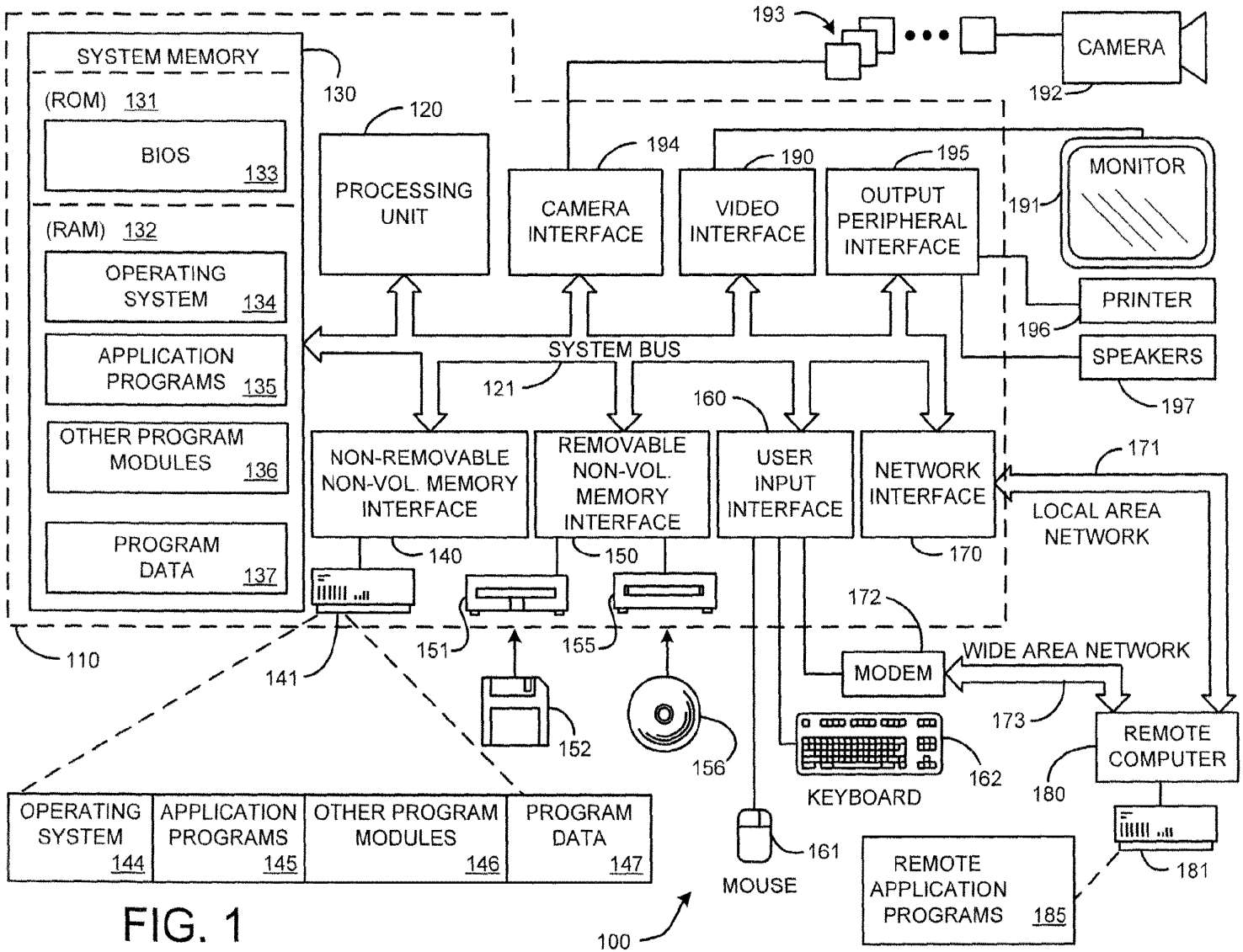

SYSTEM MEMORY
130
(ROM) 131
BIOS 133
(RAM) 132
OPERATING SYSTEM 134
APPLICATION PROGRAMS 135
OTHER PROGRAM MODULES 136
PROGRAM DATA 137
120
PROCESSING UNIT
194
CAMERA INTERFACE
190
VIDEO INTERFACE
195
OUTPUT PERIPHERAL INTERFACE
193
CAMERA
192
MONITOR
191
PRINTER
196
SPEAKERS
197
SYSTEM BUS
121
160
NON-REMOVABLE NON-VOL. MEMORY INTERFACE
REMOVABLE NON-VOL. MEMORY INTERFACE
USER INPUT INTERFACE
NETWORK INTERFACE
171
LOCAL AREA NETWORK
140
150
170
110
141
151
155
172
MODEM
WIDE AREA NETWORK
173
152
156
KEYBOARD
162
REMOTE COMPUTER
180
161
MOUSE
181
REMOTE APPLICATION PROGRAMS 185
OPERATING SYSTEM 144
APPLICATION PROGRAMS 145
OTHER PROGRAM MODULES 146
PROGRAM DATA 147
100

FIG. 1

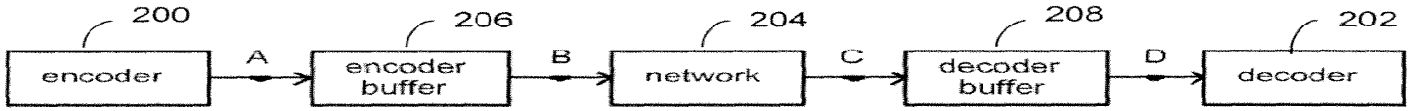


FIG. 2

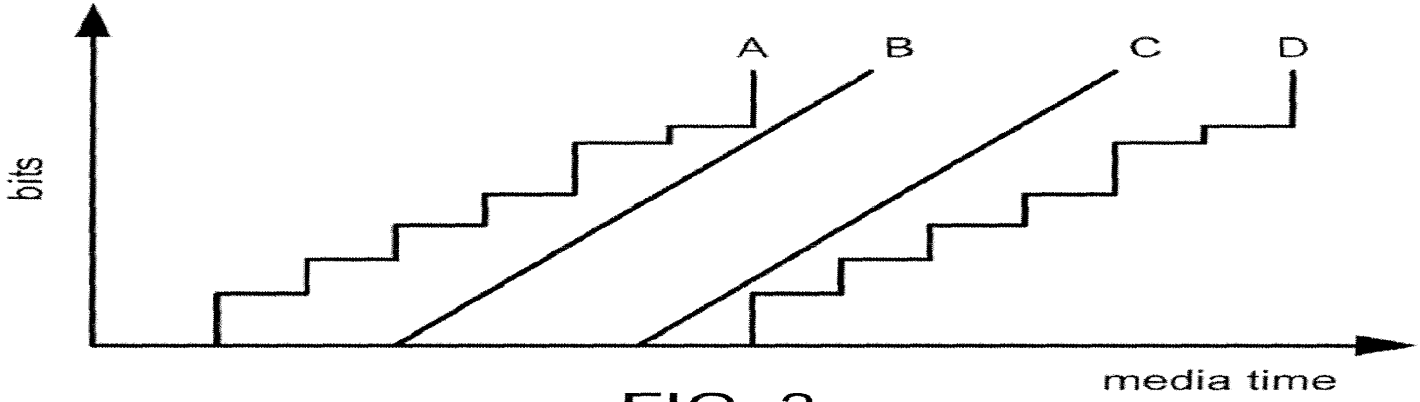


FIG. 3

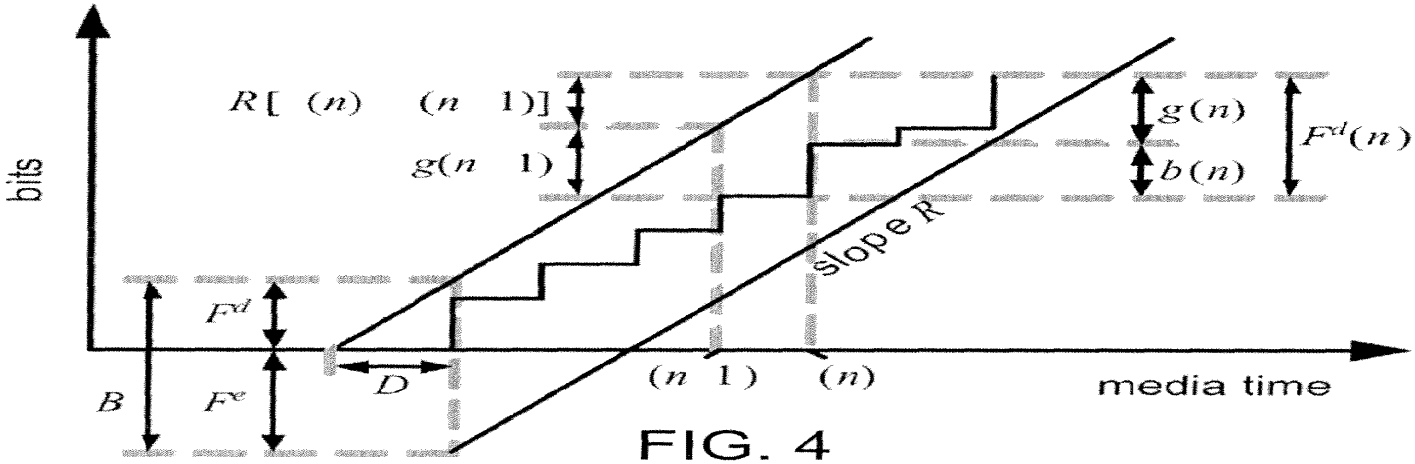


FIG. 4

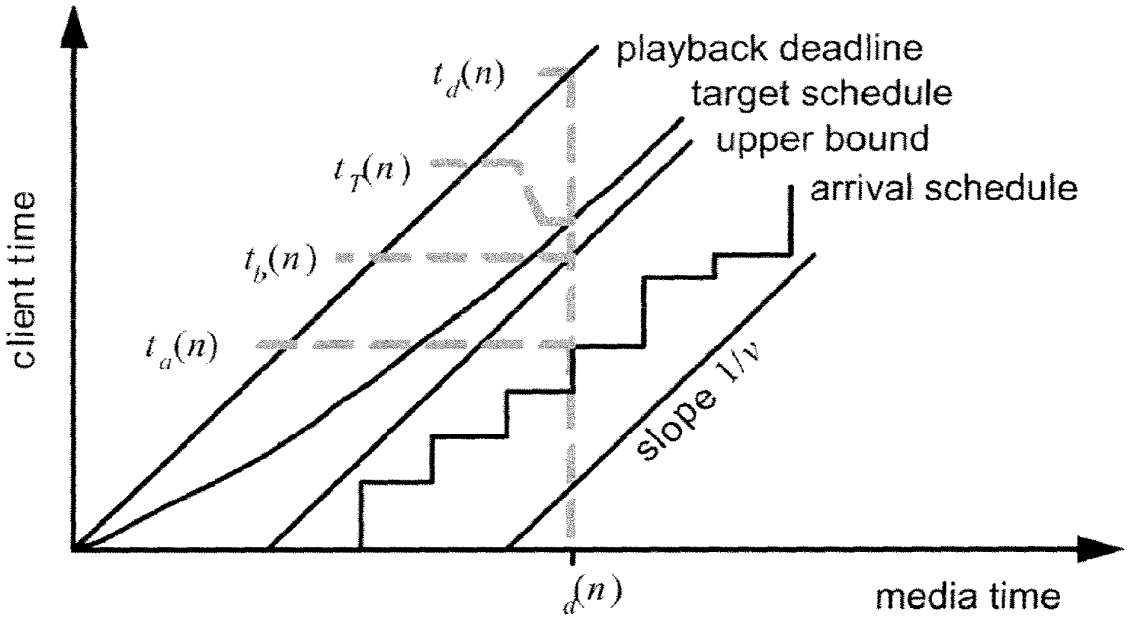

playback deadline
target schedule
upper bound
arrival schedule
$t_d(n)$
$t_T(n)$
$t_b(n)$
$t_a(n)$
client time
slope 1/v
$_d(n)$
media time

FIG. 5

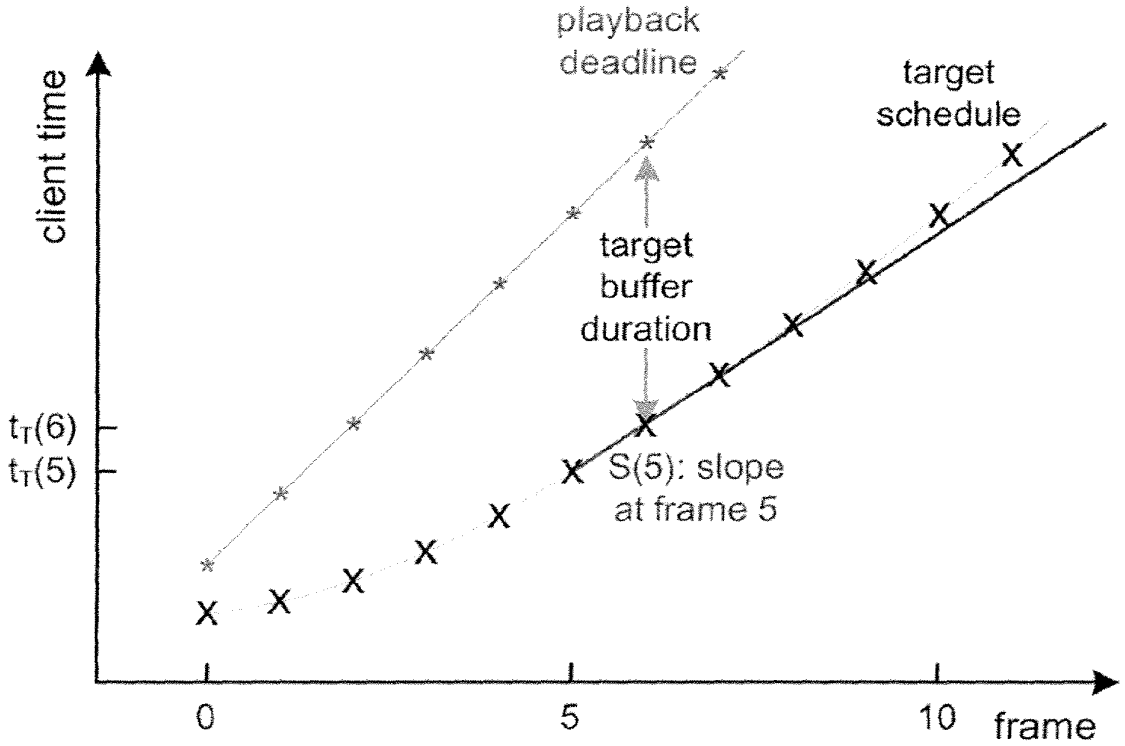

playback deadline
target schedule
target buffer duration
client time
$t_T(6)$
$t_T(5)$
S(5): slope at frame 5
0
5
10
frame

FIG. 6

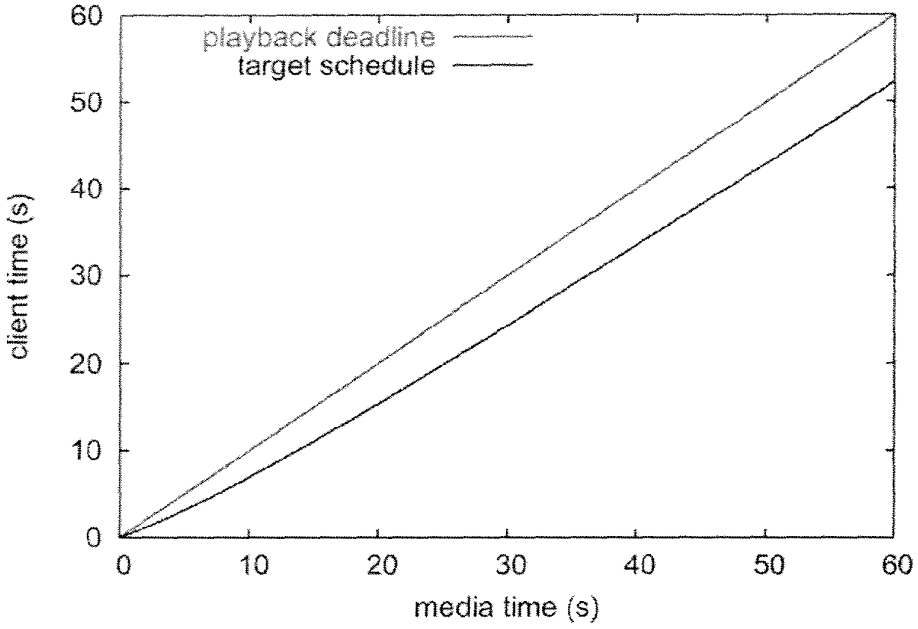


FIG. 7(a)

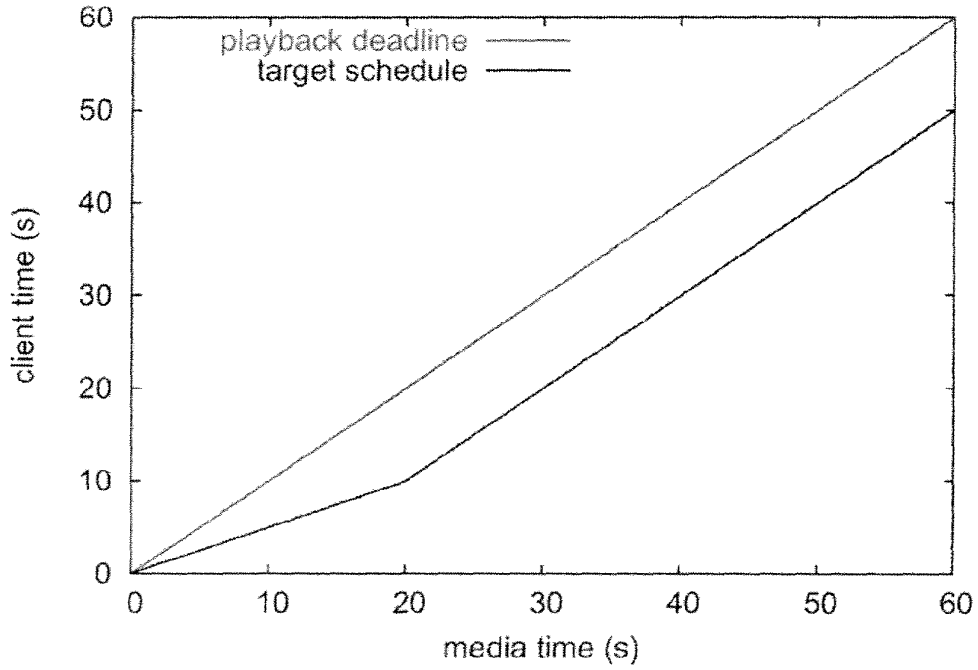


FIG. 7(b)

Aylo Exhibit 1002
Page 252 of 1405

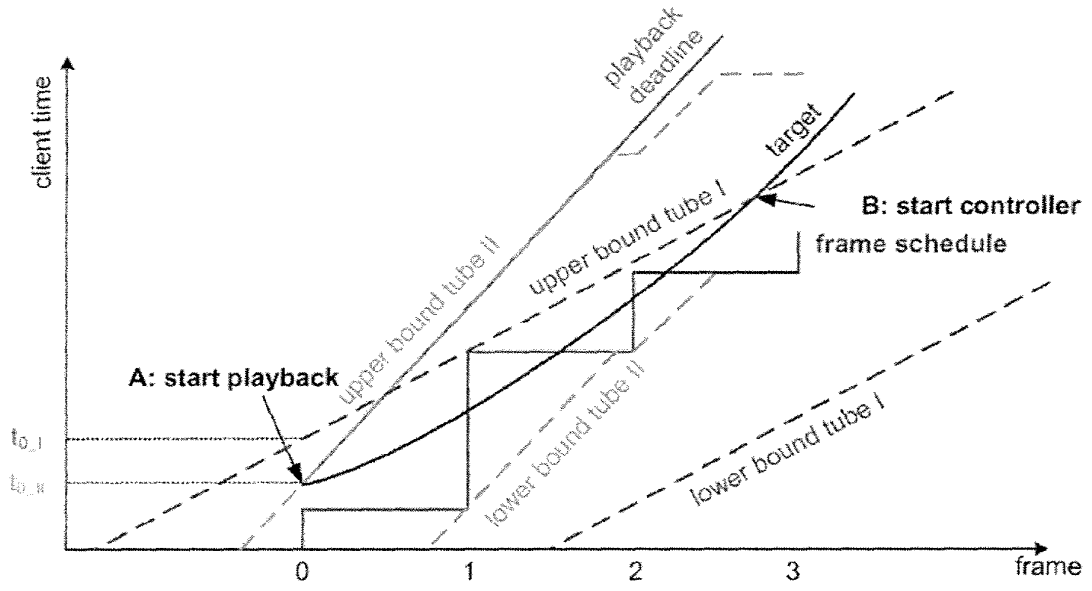


FIG. 8

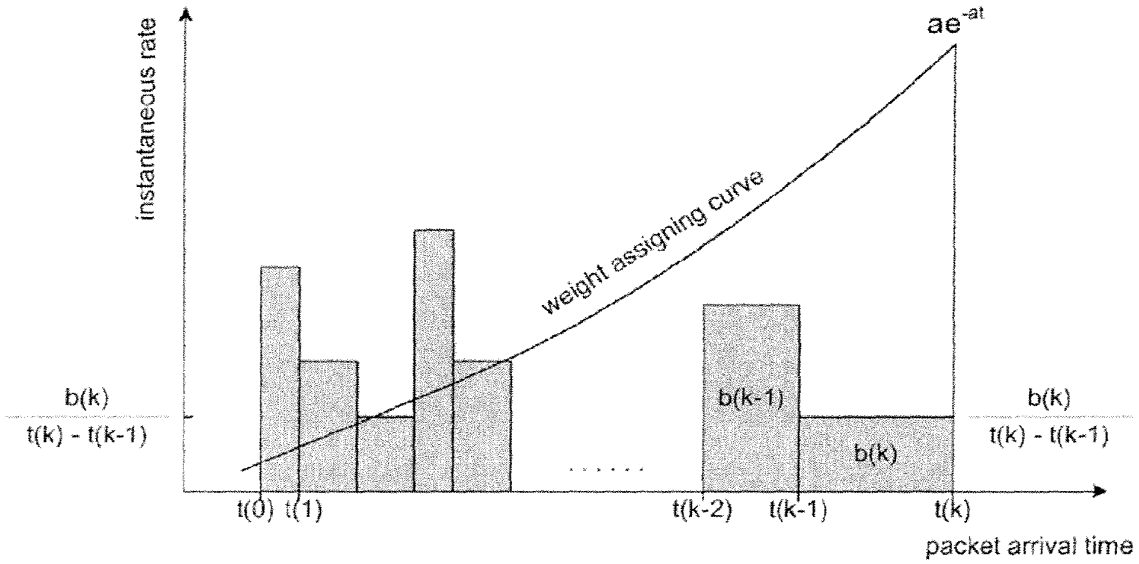


FIG. 9

Aylo Exhibit 1002
Page 253 of 1405

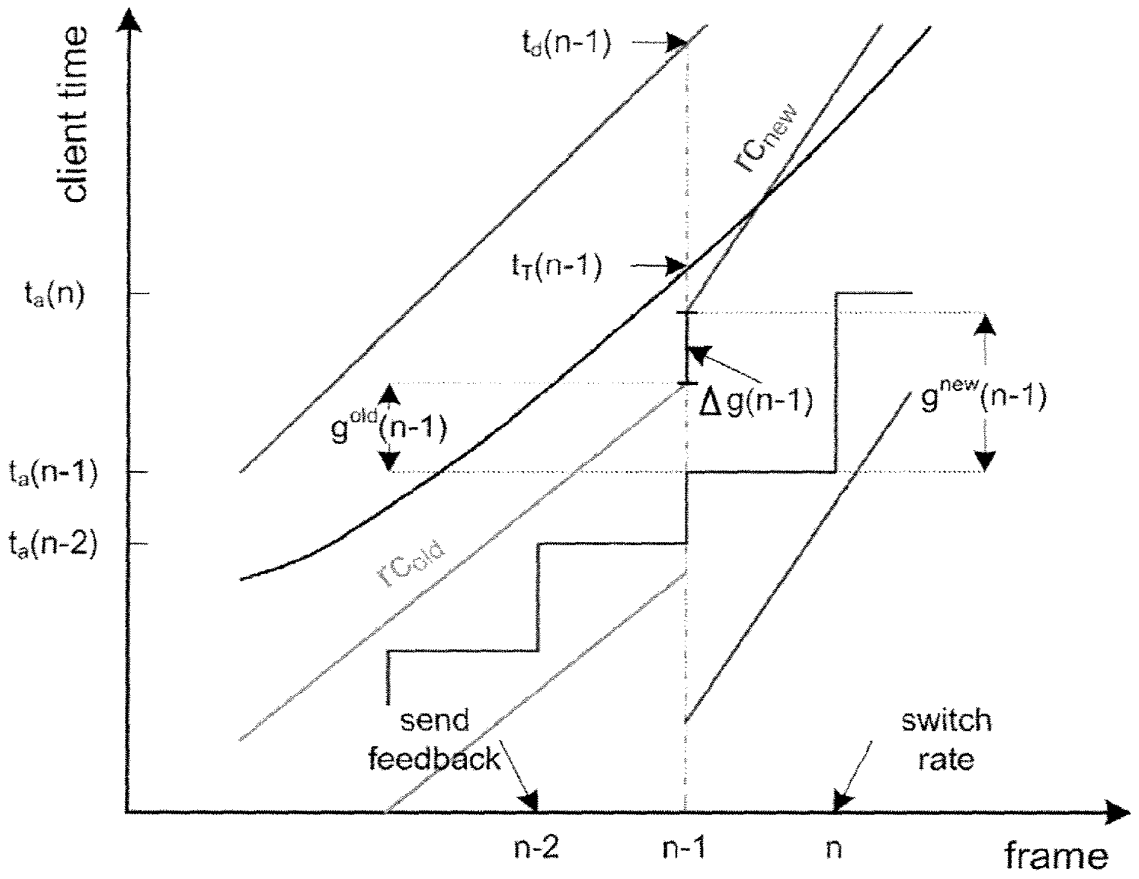


FIG. 10

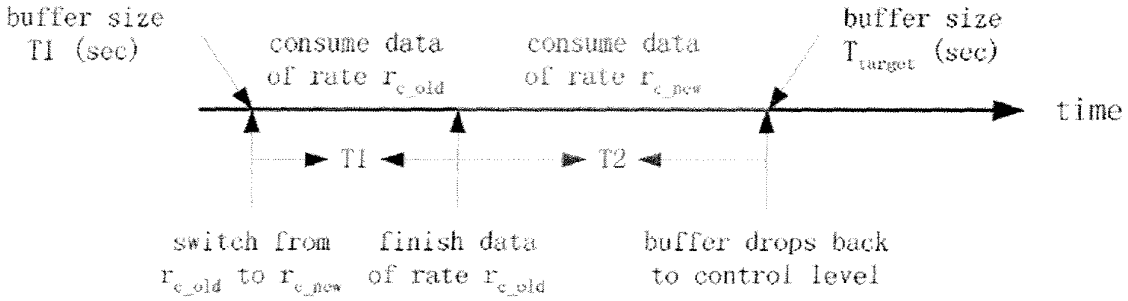


FIG. 12

Aylo Exhibit 1002
Page 254 of 1405

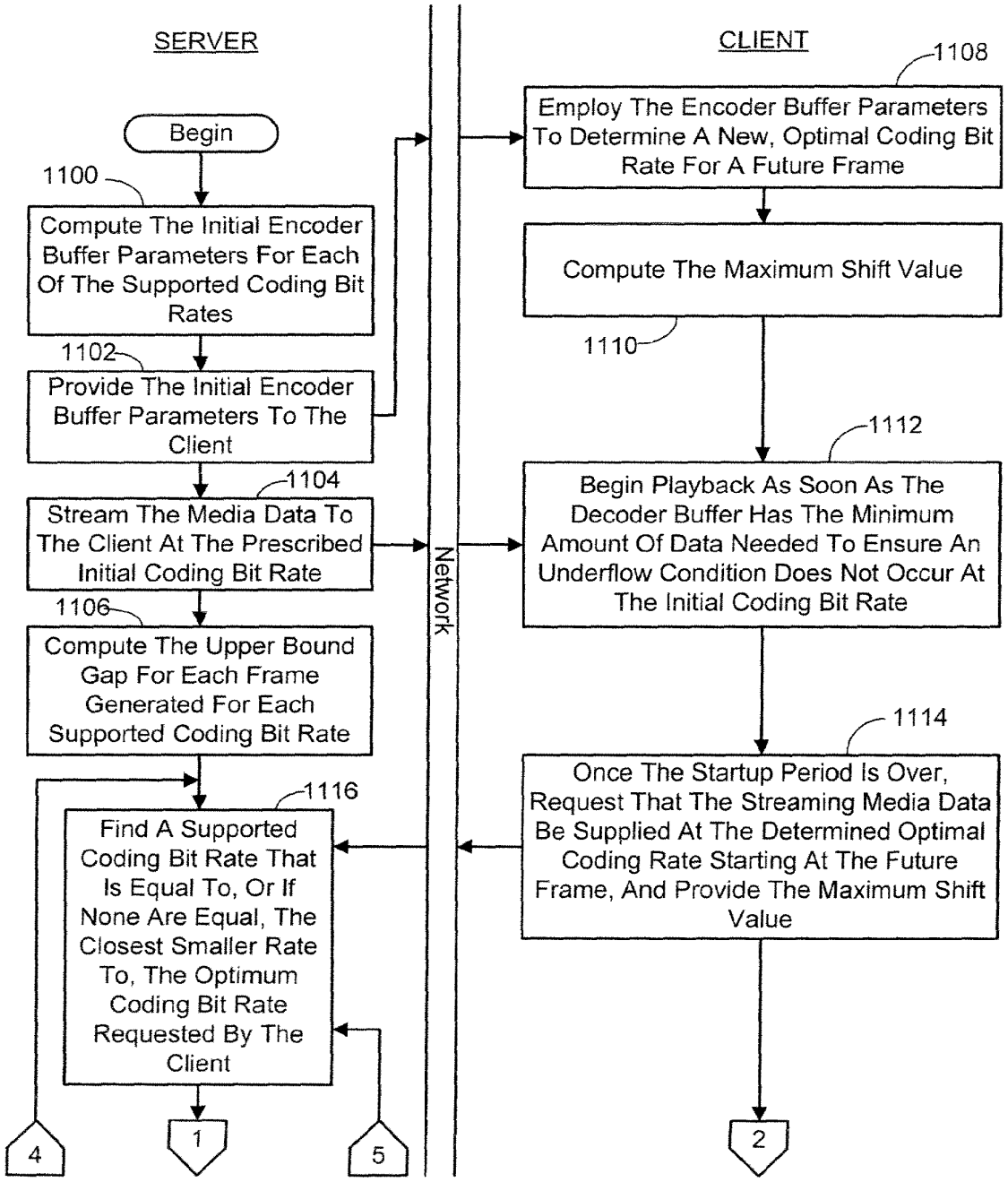


FIG. 11A

Aylo Exhibit 1002
Page 255 of 1405

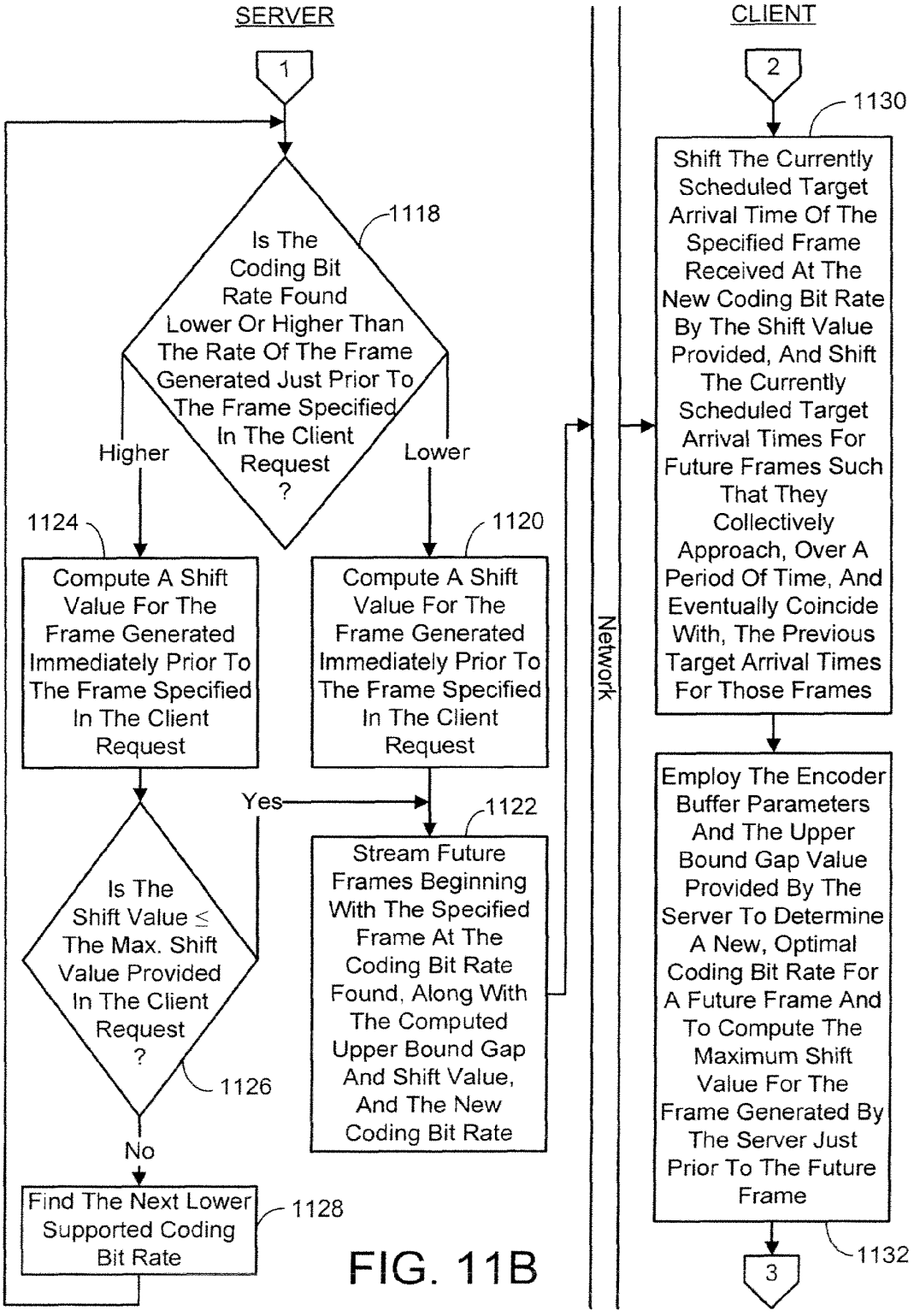


FIG. 11B

Aylo Exhibit 1002
Page 256 of 1405

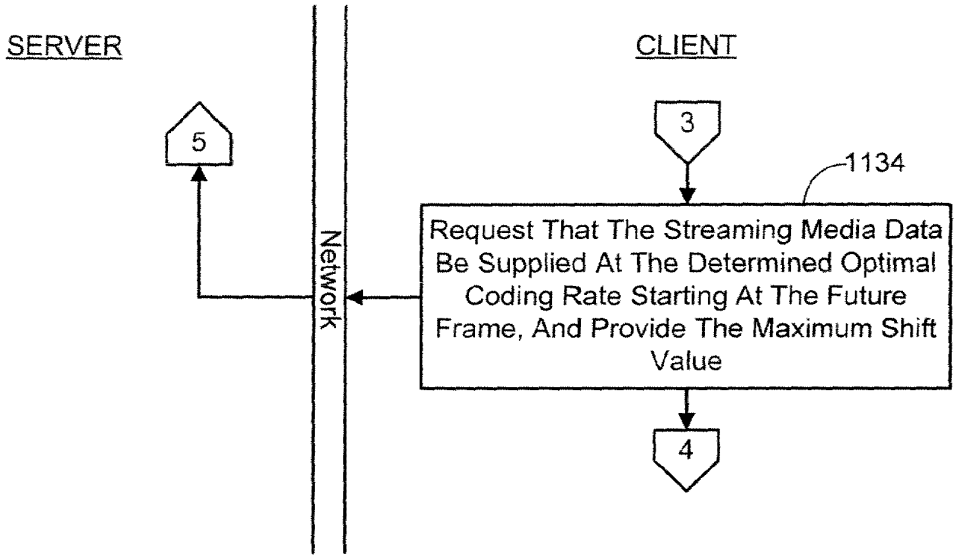

SERVER
CLIENT
5
3
1134
Network
Request That The Streaming Media Data Be Supplied At The Determined Optimal Coding Rate Starting At The Future Frame, And Provide The Maximum Shift Value
4

FIG. 11C

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

(11) EP 0 919 952 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
02.06.1999 Bulletin 1999/22

(51) Int. Cl.[6]: **G06T 9/00**

(21) Application number: **98109423.8**

(22) Date of filing: **25.05.1998**

(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: **28.11.1997 US 66856 P**

(71) Applicant:
ECOLE POLYTECHNIQUE FEDERALE DE LAUSANNE
1015 Lausanne (CH)

(72) Inventors:
- **Mattavelli, Marco**
1196 Gland (CH)
- **Brunetton, Sylvain**
1009 Pully (CH)

(74) Representative:
Ravenel, Thierry Gérard Louis et al
I C B,
Ingénieurs Conseils en Brevets SA,
7, rue des Sors
2074 Marin (CH)

Remarks:
A request for correction of the claims numbering has been filed pursuant to Rule 88 EPC. A decision on the request will be taken during the proceedings before the Examining Division (Guidelines for Examination in the EPO, A-V, 3.).

(54) Method for coding/decoding of a digital signal

(57) The invention concerns a method for primary processing a digital signal comprising the steps of :

- providing a digital signal to a primary processing unit ;
- primary processing said signal with said primary processing unit according to a plurality of primary processing algorithms to provide primary processed output signals,
- determining statistics of use of at least one of said primary processing algorithms,
- providing a digital statistics signal representative of said statistics for each of said primary processed output signals
- associating to each of said primary processed output signals its own statistics signal ;
- determining a complexity primary processing information signal based on said statistic signal.

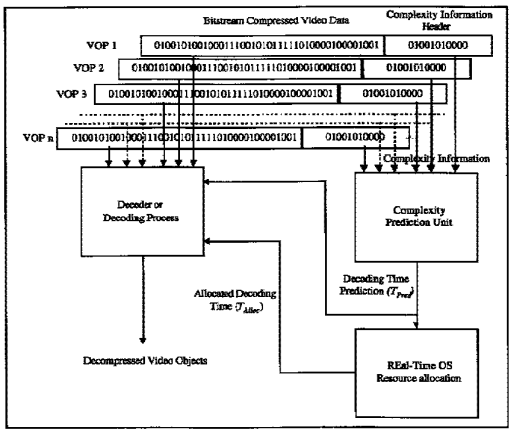


Figure 12

EP 0 919 952 A1

Aylo Exhibit 1002
Page 258 of 1405

Description

DEFINITION OF THE INVENTION

[0001] The present invention is a processing apparatus which aims at controlling and guaranteeing the real-time performance of data processing. The invention is based on the principle of collecting statistics of the algorithms used to process the data, on transmitting such statistics when necessary, and of associating a complexity information to such statistics. The real-time processing control can be done generating a processing time prediction signal which is a function of the complexity information, and use such prediction signal to provide processing resources which guarantee real-time processing or enables the definition of an associated degraded processing that guarantees real-time processing. Although the described invention is of general application, this document describes its application to video coding and decoding and suggests the straightforward extensions to audio coding-decoding, and to composition and rendering of media objects.

1. FIELD OF THE INVENTION

[0002] The implementation of software based video/audio decoders is an advantageous solution over traditional dedicated hardware systems. The main reason of this fact is the possibility of decoding several different video/audio compression standards, at different levels and profiles, by just changing the software while using the same hardware platform, main memory, and interfaces. Another favorable feature is the flexibility of the decoder able to cope with new developments and new tools included into future versions of the standards by just updating the decoding software. However, simultaneously guaranteeing real-time performance, needed when processing video/audio bit-streams, and an efficient use of the processing power, remains a very difficult task.

[0003] The present invention reports the main reasons of such difficulties and presents a new technique able to predict the decoding time of compressed video/audio bit-streams without the need of the actual decoding. This can be obtained if appropriate statistical information about the actual encoding processing is included and transmitted with the compressed video/audio data. Moreover, the present invention reports how such results can be used to implement new efficient processing resource allocation strategies. New possible schemes of such intelligent interactions between the encoder and the real-time OS are also proposed. An example of implementation for the MPEG-4 VM6 video algorithm based on these developments is presented. This implementation enables, at the same time, to guarantee real time performances even when the available processing resources are lower than the one theoretically necessary, and it yields optimal (minimal) image quality degradation. Such results are obtained by using the prediction of the decoding-resource needs to implement Computational Graceful Degradation techniques. The invention only reports simulation results for the MPEG-4 VM6.1 video compression algorithm, but the techniques and concepts presented here are of general application. Indeed, these techniques can be easily applied to audio coding, synthetic audio reproduction and to any future or existing compression and composition method. More precisely, they can be used for any video compression standard that is capable of transmitting in an appropriate form the statistical coding information (i.e. MPEG-1 and MPEG-2 using an appropriate transport system), and they can be implemented in any processor-based platform.

[0004] The present description is organized as follows. Section 2 discusses the problems and utility of defining and using worst case and average case complexities, justifying the need of complexity prediction. Section 3 presents the modeling of decoding complexity and the general principles of the prediction techniques. Section 4 presents the background of the invention. Section 5 summarizes the main concepts of the invention while section 6 summarizes the content of the diagrams and drawings illustrating the details and main characteristics of the invention.

[0005] In section 7, experimental results of the MPEG-4 VM6 texture coding algorithm are reported. Section 8 presents the definition of the proposed intelligent interactions between real-time OS-schedulers and video decoders. Section 9 presents the basic principles and the definition of the developed Computational Graceful Degradation (CGD) techniques, and section 10 reports overall experimental results of the control of decoding complexity using complexity prediction joint with the application of CDG techniques. Finally, section 11 concludes the description of the invention, summarizing the achievements of the invention by declaring the corresponding claims.

2. WORST-CASE AND AVERAGE-CASE DECODING COMPLEXITY

[0006] A simple solution to the problem of the variability of decoding complexity is to implement software/hardware solutions that always guarantee worst case complexity decoding. Such solutions are, in fact, the ones implicitly implemented in most of the dedicated hardware decoders. Dedicated hardware platforms, in general, are designed to guarantee worst case complexity decoding. The various hardware modules implementing the different decoding procedures are synchronized with the input/output data rate. These dedicated hardware modules can always handle any input bit-stream, not exceeding a maximum rate, and containing compressed sequences of appropriate maximum image reso-

Aylo Exhibit 1002
Page 259 of 1405

lutions. This can be always guaranteed, irrespectively from the coding options selected to produce the compressed bitstream.

[0007] In contrast, in the case of programmable platforms it is necessary to define what is a worst case decoding complexity. Let us consider video coding even if the same concept can be extended to natural and synthetic audio and to other form of multimedia processing. In the video compressing field one can assume that the worst case decoding complexity is represented by the decoding of each bit/macroblock, using the coding options allowed by the considered standard that requires the highest decoding time. The coding algorithms/modes of H.263 can be approximately considered as a superset of the algorithms/modes of the various video compression standards (H.261, MPEG-1 and MPEG-2). The complete results and details of this worst-case/average-case study are omitted here for brevity. The conclusions resulting from this analysis can be summarized as follows :

1) Decoding functions can be divided into two classes. For one class a worst case can be easily defined. Let us denote this class as A. To this class belong function such as IDCT, inverse quantisation, motion compensated prediction in its various form (forward, backward, bi-directional, overlapped ... .). For the other class denoted as class B, complexity is extremely data dependent and cannot be easily characterized by the simple parameters of an hypothetical MPEG-2 like complexity profile. To this class belong operations such as VLD, VLC coding, Syntax-based Arithmetic Coding (SAC) and in general all parsing operations. The difficulty of defining a worst case is evident considering for instance SAC for which the difference between a theoretical worst-case and experimental cases is about two orders of magnitude.

2) A strict worst case complexity analysis, of both class A and B functions, leads to unrealistic results. In other words, the computational resources necessary to handle the strict worst-case complexity decoding is much higher than the resources used in average.

3) For class A functions, the actual decoding complexity depends on the actual coding options and reports a large range of variability.

4) For class B functions, the actual decoding complexity reports a dependence on coding options, but presents a reasonably narrow range of variability which is much lower than the range theoretical worst case-average case.

[0008] Worst case analysis, therefore, is not adequate to define a useful decoding complexity reference. This fact is true not only for operations which are strongly data dependent (class B), but in general for all functions even those that have a very well defined worst case such as all class A functions. From these results it is clear that guaranteeing a worst case decoding implies guaranteeing the decoding of pathologically complex sequences that in practice have an extremely low probability of occurrence. Moreover, such a guarantee would imply the allocation of a processing power up to two orders of magnitude higher than the one actually needed for the decoding of the standard test sequences. The processor in practice would work in normal conditions exploiting only a small fraction of the theoretically available processing power. Such solution to the problem of guaranteeing real-time performance is not economical and, obviously, not useful in practical applications.

[0009] Another possibility for the solution of the problem of guaranteeing real-time performance could be to consider as reference an average case decoding complexity. Before being able to successfully use such average complexity, a few questions need to be answered. Is it meaningful to define an average complexity? How can it be done? Is the average complexity sequence dependent? How can it be converted on different platforms? Can the average complexity accurately be described by parameters such as image resolution and bit-rate? Are average results characterized by sufficiently narrow variance bounds so as they can be used a priori without the need of knowing some information about the actual complexity of the incoming sequence?

[0010] One will here briefly summarize the main results and experiments that answer the mentioned questions. So as to try to define an average case for the complexity that have been considered in previously, the H.263 standard uses four typical test sequences over a wide bit-rate range from 16 kbit/sec up to 1024 kbit/sec. The sequences in CIF format are : Akiyo (300 frames), Coastguard (300 frames), Foreman (300 frames) and Mobile & Calendar (300 frames), and all these test sequences have been evaluated at fifteen different bit-rate values. Although a rigorous statistical analysis would have required a much higher number of sequences and specific statistical confidence tests, it is believed that the number of frames (1200) and the different contents of the test images are well representative of typical image sequence contents. One have considered separately the operations that have unrealistic worst case complexities (class B) from the one with a well-defined worst case behavior (class A). The experiments, for the class B operations, yield figures that are not strongly bit-rate dependent. Variance bounds for each frame are relatively narrow and these values do not report relevant variations depending on the test sequence. Such results confirm also the fact that it is not necessary to worry about strict worst case complexity. In the other case of class A, the complexity of the operations reports a relatively regular statistical behavior at very low bit-rates (< 16 kb/s), but complexity reports dependence on the bit-rate, on the image content and in particular on the used encoding options.

[0011] Figure 1 reports one example of H.263 complexity using different coding options for the same sequence (Akyio

Aylo Exhibit 1002
Page 260 of 1405

CIF) and same bit-rate. On this figure, the decoding time or decoding complexity (expressed in μsec) is given versus the frame number for the sequence Akyio CIF encoded with H.263 at 128 Kb/sec using different coding options. Top curve options : advanced prediction modes (APM), syntax-based arithmetic coding and unrestricted motion vectors (UMV) ; middle curve options : APM and UMV ; bottom curve options : UMV. The same sequence presents, at the same bit-rate, average-decoding complexities that range up to a factor 4 (from about 0.05 to 0.2 sec per frame). Local fluctuations for a relatively static sequence such as Akyio are also considerably large. The decoder is the Telenor H.263 software release running on a Sun Ultrasparc platform.

[0012] Figure 2 reports the same experiment for the sequence Coastguard. The range of the complexity figures considering both average and local values is very large as can be easily noticed. On this figure, decoding time or decoding complexity (expressed in msec) is given versus frame number for the sequence Coastguard encoded with H.263 at 128 Kb/sec using different coding options. Top curve : advanced prediction modes (APM), syntactic arithmetic coding and unrestricted motion vectors (UMV) ; middle curve APM and UMV ; bottom curve : UMV. The same sequence at the same bit-rate has average decoding complexities that range within a factor 4 from about 0.05 to 0.2 sec per frame. The decoder is the Telenor H.263 software release on a Sun Ultrasparc platform.

[0013] Although far from the theoretical worst case complexities, the variation range is less than one order of magnitude, experimental results reports behaviors for which it is difficult to be able to predict the average complexity values only relying on resolution and bit-rate. Image content and coding options (i.e. intra or inter prediction modes, advanced prediction modes, half pixel vector prediction and so on) play a fundamental role. As it will be reported in the next sections, the situation of the complexity variations for MPEG-4 sequences is much higher due to the presence of different sized VOPs arbitrarily shaped objects, of static and dynamic sprites, and the large variety of coding modes.

[0014] From a pure mathematical point of view an average complexity value can always be defined considering a large data-base of encoded sequences and using the more probable occurrence of encoding options. Conversely, such a theoretical average value is not useful at all for the aim of the present invention that is the efficient scheduling of each decoding task and the efficient use of the available processing power. Even the local average values of the actual sequence often are not stationary, and their variations around the mathematical average one are so unpredictable that no useful scheduling policy can be realized.

[0015] Therefore, the consequences of this study are that straightforward solutions to the problem of real-time resource allocation policy are of two kinds. The first is a sustained over-sizing of the processing resources. The second is to accept interruptions of the audio/video output stream when the needed decoding resources exceed the ones available on the system. Obviously, both are not desirable being respectively not efficient and economical, or they provide a quality of the services which may easily results perceptually unacceptable.

[0016] The ideal solution would be to develop techniques that are able to accurately predict such unknown behaviors and, therefore, make feasible resource allocation policies that can guarantee real-time and minimize QoS degradation.

[0017] The description of such techniques and the presentation of the results are the subjects of the next sections.

[0018] As discussed above, the key issue for the implementation of any real-time application on a software-based platform is to be able to understand when the required decoding processing time exceeds the available processing time without the need of performing the actual decoding. This is particularly important, for instance, in MPEG-4 systems in which a number of bit-streams containing a variable number of VOPs and sprites need to be decoded respecting real-time constraints, but it is also important for existing compression standards (MPEG-1, MPEG-2). The available processing time is known, in general, because it is allocated by the OS-scheduler to each decoding process. The problem is how to estimate the necessary decoding processing requirements without the need of an actual decoding of each of the incoming VOP bit-streams. Decoding each bit-stream using a priori allocated processing could result in exceeding the time slot and, therefore, in missing real-time requirements. Real-time requirements are generally defined implicitly by the frame/sampling rate of video audio bit-streams or can be easily extracted by an appropriate relative time information associated to each video frame or audio object (also denoted as time stamp). Such extraction of the real-time constraints is trivial and is not discussed here. The consequence of missing real-time constraints is the introduction of sudden interruptions on the video content. A complete analysis of the conditions which lead to miss real time-constraints for the various OS scheduling algorithm (pre-emptive multi-threading, etc.) is out of the scope of this document and it is not discussed here. The reader can refer to the included reference S. Battista, F. Gasperoni, B. Klaiber, M. Mattavelli, L. Pautet, S. Tardieu, Concurrent over Sequential Modeling for MSDL, ISO/IEC JTC1/SC29/WG11, Doc. M1525, Maceio, Brazil, November 1996 incorporated herein by reference.

3. BACKGROUND OF THE INVENTION

[0019] Without any doubt, the recent trend in multimedia system implementation technology is towards moving from dedicated real-time hardware to software/processor based platforms. The processing power of the last on the market processors, and of the newly announced video/audio-processor families, enables the software implementation of video/audio decoding of various compression standards and different resolution levels. Unfortunately, simply relying on

Aylo Exhibit 1002
Page 261 of 1405

the average processing power does not guarantee to respect real-time constraints of video/audio decoding ; furthermore it does not solve the problem of an efficient processing resource allocation. The fundamental reason of these facts is that the complexity of the video/audio decoding process is a fluctuating function of the time, as described in M. Mattavelli and S. Brunetton, A statistical study of MPEG-4 VM texture decoding complexity, ISO/IEC JTC1/SC29/WG11, Doc. M924, Tampere Finland, July 1996 incorporated herein by reference.

[0020] The term complexity here simply means the time needed by the decoding process. Other more abstract measures of complexity such as the amount and type of executed algorithmic elementary operations, the executed processor cycles, the amount of memory transfers to and from the external frame memories, and their combinations could be used as more precise and specific complexity measures. In the context of the present invention however, one is interested on the final results of these complexity factors. Without any loss of generality, one can consider, for any specific target software-hardware implementation, the actual processor decoding time. Indeed all the above mentioned measures of complexity can be converted for each specific platform into a single complexity measure given by the execution time.

[0021] Even by considering a specific processor platform or a specific compression standard with a relative level and profile, the complexity of video decoding is a variable and unpredictable quantity, as described in ISO/IEC MPEG2 incorporated herein by reference. The actual figures of decoding complexity depend in a complex way on many different factors. One can mention, for instance, the actual bit-rate, the image content, the options used for encoding, the Group Of Picture (GOP) structure, or the Group Of Video Object Planes (noted as VOP) structure (GOV) which can change on the fly, the state of the buffer that controls the quantization parameters and many others, as described in M. Mattavelli and S. Brunetton, Scalability of MPEG-4 VM based on the computational power of texture decoding, ISO/IEC JTC1/SC29/WG11, Doc. M925, Tampere, Finland, July 1996 incorporated herein by reference.

[0022] The complexity variability range increases with the richness of the standard. Richness means the number of implemented coding algorithms or coding modes and the number of possible coding options having different complexities. For instance, the richness of H.263 is much higher than H.261, but it is lower than the new MPEG-4 standard under development where multiple Video Object Planes (VOPs) of different sizes and characteristics need to be decoded.

[0023] In this description, Video Object (VO) means a temporal sequence of two-dimensional or three-dimensional arbitrarily shaped Video Object Planes (VOP). In case of rectangular or square shapes, these video object planes, also called hereinafter video planes, are denoted by the classical name of frames. An audio object means a stream of packets containing digital audio samples, or audio events in mono or multichannel format which can be associated to a VO. One or more audio packets can be associated to one VOP or video frame. By three-dimensional arbitrarily shaped video object planes we mean the projection or view of any three-dimensional natural or synthetic 3-dimensional object or 3-dimensional model.

In summary, the processor decoding time of each incoming frame or VOP for any given compression standard is highly variable and impossible to predict just relying on resolution and bit-rate information. Therefore, an efficient resource allocation policy (process scheduling) which aims at guaranteeing real-time performances is very difficult in these conditions. Scheduling policies that release the thread to the decoding process and receive it back after the task is completed are clearly not adequate. The same conclusion is true for a priori allocated processing time intervals. If more processing time than the one actually allocated is needed, real-time constraints are already violated by the video application. The implementation of emergency measures such as skipping frames or portions of images, on one hand might succeed in recovering the correct timing, on the other could have catastrophic effects on the Quality of Services (QoS) of the video application.

4. SUMMARY OF THE INVENTION

[0024] It is an object of the present invention to control and to guarantee real-time performance of data processing. It is another object of the present invention to provide a processing apparatus that associates a complexity information to the processed data.

[0025] It is another object of the present invention to provide a coding-decoding apparatus that further includes the associated complexity information to be transmitted to the decoder.

[0026] It is another object of the present invention to provide a system that, on the basis of said complexity information, provides a complexity prediction signal to be used to allocate processing resources to optimize real-time data processing.

[0027] It is another object of the present invention to provide a processing system that supervises the processing using an allocated resource signal, a predicted complexity signal, a complexity information signal and provide a processing that satisfy specified real-time constraints. According to the present invention, there is provided a processing apparatus comprising a method for processing a digital signal including the steps of : providing a digital signal, processing said signal using a plurality of processing algorithms, determining statistics of use of processing algorithms, providing a digital statistical signal representative of the statistics, associating to each of the processed signal its own

Aylo Exhibit 1002
Page 262 of 1405

associated statistics and determining a complexity information signal based on said statistic signal.

[0028] According to the present invention, there is also provided a coding apparatus that further inserts the said complexity information signal into a transmission layer containing or not the associated data and transmits the said complexity information to a decoder apparatus.

[0029] According to the present invention, there is also provided a processing apparatus that generates from the complexity information signal a processing prediction signal which is used to allocate processing resources in order to optimize the real-time processing of the associated data.

[0030] According to the present invention, there is also provided a processing apparatus that receives the processing prediction signal, the allocated processing resources and the complexity information signal, and uses the said signals to generate processing directives so as the data is processed according to the system real-time constraints.

[0031] Although the described invention is of general application, the present description describes its application to the preferred embodiment of the video coding and decoding, and suggests the straightforward extensions to audio coding-decoding and to composition and rendering of media objects.

[0032] The technique provided by the present invention is capable of predicting with an excellent degree of approximation the complexity of video or audio decoding, without the need of the actual decoding of the incoming bit-stream. These results are possible when an appropriate coding information can be included in the incoming bit-streams. The experiments demonstrate that the degree of approximation achievable is excellent considering the range of variability of the actual decoding complexity of video or audio compression standards, and that such degree is more than adequate to enable an efficient processor resource allocation. One application of the proposed technique to the MPEG-4 VM6 video coding algorithm is also demonstrated. Real-time video performances are guaranteed using the predicted decoding complexity to implement Computational Graceful Degradation (CGD) features whenever the video decoding complexity exceeds the available processing resources. By means of the proposed technique, it can be guaranteed at the same time the satisfaction of the real-time constraints and the aim of minimizing the unavoidable degradation of the QoS due to temporary lack of processing resources or to peaks of the intrinsic video decoding complexity. The same approach can obviously be used for audio decoders.

[0033] It is clear that the approach is not limited to the coding/decoding process for compression purposes, but that it can be applied to the process of composing and rendering media objects (i.e. video, audio, text, graphics, two-dimensional and three-dimensional graphic primitives and models) to the display output. The term composition means the action of putting these objects together in the same representation space. The action of transforming the media objects from a common representation space to a rendering device (speakers and viewing window) is called rendering. An illustrative example of the terminal environment for the handling of multimedia object is reported in Figure 19. More specific possible examples of the decoding, composition and rendering processes are reported in Figure 20 and 21. Figure 19 reports the processing stages in an audiovisual terminal illustrating inputs outputs and processing of audiovisual objects. Figure 20 reports an example of a decoder which provides uncompressed audiovisual objects to the audio/video compositor. Figure 21 reports an illustrative example of composition of audiovisual objects.

[0034] Additional objectives and advantages of the invention will be set forth in the description that follows, and in parts will be obvious from the description, or may be learned by practice of the invention. The objectives and advantages of the invention may be realized and obtained by means of the instrumentalities and combinations particularly pointed out in the appended claims.

5. BRIEF DESCRIPTION OF THE DRAWINGS

[0035] The accompanying drawings constitute a part of the specification and illustrate the principles of the invention.

- Figures 1 and 2 report a graphical representation of the decoding complexity of H.263 video illustrating the needs and background of the invention for the video decoding embodiments ;
- Figure 3 reports the an example of the modeling of processing algorithms. It includes the association of processing statistic to each state associated to the execution of a specific algorithm ;
- Figure 4 reports the extension of the model of figure 3 for a more complete and accurate model of the algorithmic complexity ;
- Figure 5 reports one preferred embodiment of the invention for the MPEG-4 video decoding. It illustrates a complexity model with the associated statistics for MPEG-4 texture decoding ;
- Figure 6 extends the model of figure 5 including additional coding modes and coding parameters plus the associated statistics ;
- Figures 7 and 8 report simulation results of a preferred embodiment of the invention. By associating a complexity measure to the statistics of the data processing (video decoding in the preferred embodiment), it is possible to predict the processing complexity of data ; actual decoding time, predicted decoding time and prediction error are represented.

Aylo Exhibit 1002
Page 263 of 1405

- Figure 9 reports the measured decoding time and Decoding Time Prediction (expressed in μsec) versus the sequence VOP number of the MPEG-4 VM6 for the sequence Coastguard (Background, Small boat, Big Boat and Water VOPs) at 64 Kb/s ; these results being obtained on a specific decoder.
- Figure 10 reports the measured decoding time and Decoding Prediction Time (expressed in μsec) versus the sequence VOP number of the MPEG-4 VM6 for the sequence Coastguard (Background, Small boat, Big Boat and Water VOPs) at 128 Kb/s ; these results being obtained on a specific decoder.
- Figure 11 reports the measured decoding time and Decoding Prediction Time (expressed in μsec) versus the sequence VOP number of the MPEG-4 VM6 for the sequence Coastguard (Background, Small boat, Big Boat and Water VOPs) at 128 Kb/s ; these results being obtained on a specific decoder
- Figure 12 reports a block diagram of part of the invention. The statistics carrying the complexity information and associated to each data to be processed are read by a prediction unit which generates a prediction signal sent to the OS scheduler which allocates resources as function of the prediction signal. The figure is reporting one example of preferred embodiment for the video decoding of multiple VOP ;
- Figure 13 reports a block diagram of part of the invention relative to the construction of the complexity information header at the encoder side. Statistics of the coding and/or decoding algorithms/modes/parameters are generated at the encoder side and then inserted in a bit-stream to be transmitted to the decoder ;
- Figure 14 reports a block diagram of part of the invention relative to the interaction between the decoder supervisor and the decoder using various signals in order to guarantee real-time decoding of the data ;
- Figure 15 reports the diagram of a complexity model including associated degradation algorithms ;
- Figures 16, 17, 18 report examples of results of a preferred embodiment of the invention. The video decoding for the MPEG-4 texture decoding is provided imposing a maximum constant decoding time (complexity) for each VOP. Degradation results are also reported ;
- Figure 19 reports an example of a possible preferred embodiment of the invention illustrating the various interactions and processing related to the handling of media objects (audio, video, text, graphic, 2-D and 3D models) in a multimedia terminal ;
- Figure 20 reports an example of a portion of multimedia terminal block diagram that is another possible embodiment of the invention, and
- Figure 21 reports a drawing illustrating the composition and rendering processes, both being possible preferred embodiments of the invention.

6. PREDICTING DECODING COMPLEXITY BY MEANS OF ENCODING STATISTIC INFORMATION

6.1 A model of decoding complexity for data dependent processing

[0036] To achieve the objectives of the present invention, one need a simple model of data processing complexity that preferably has the following properties :

1) it is only related to the processing of the incoming data,
2) it is as compact as possible so that its description requires the minimum amount of information (number of bits),
3) it is representative in terms of processing complexity,
4) it is not related to the any specific software or hardware implementation of the given processing,
5) it can be easily mapped on any software/hardware platform implementation,
6) it is sufficiently accurate to yield useful predictions of the processing execution time.

[0037] A way of describing the operations of processing incoming data can be to consider the sequence of ensembles of operations as appropriate distinct algorithms. If the sequence of algorithms is not dependent explicitly on the actual content of the data to be processed or by the intermediate results of the processing, the complexity of the processing can be easily calculated a priori and the complexity-modeling problem is trivial.

[0038] In general, if the sequence of operations is explicitly data dependent (i.e. the processing instructions are contained in the data themselves) and depends also on intermediate results of the processing, the problem of modeling algorithmic complexity is much more difficult.

[0039] It is well known that any processing can be modeled by a finite-state-machine for which next state depends on the input, current state and current output (the well-known machine of Mealy). This formulation is rigorous and general. In theory it could also be used to measure the processing complexity. If one assume to associate a complexity measure to each state transition, one could measure the global complexity of the processing by monitoring the state of the machine at each step, and adding up the complexity related to each state transition. In practice, such model is not useful for two main reasons. The state machine description of usual video decoding would result extremely complex. The global complexity associated to the sequence of state transition would result too abstract to make feasible the mapping

Aylo Exhibit 1002
Page 264 of 1405

on different implementations than the canonical state machine. In practice, it would be extremely difficult to derive a complexity expressed as total execution time for different specific platforms.

[0040] A more useful model is to represent the decoding process by a flux diagram for which each data state corresponds to the execution of a specific algorithm f. Such representation is more naturally related to the usual representations of the execution of software programs. Let us associate to each data state a measure of the occurrence of such state, which is incremented each time the data processing falls in each state. Let us suppose to monitor the execution of the decoding process and, for the occurrence of each algorithm execution, to increment the associated measure. The result of this model is a representation of the data processing given by a sequence of measures associated to each algorithm composing the complexity model. In general, any encoder knows which algorithm sequence is needed for the decoding, since the encoding and the coding syntax define without ambiguities the decoding algorithms . In particular, video decoders have to implement a decoder, to build, reference images, therefore they are able indeed to generate the described complexity representation.

[0041] This modeling approach has the nice feature that admits different levels of complexity resolution. One can consider only a very limited number of main algorithms, or partitioning into a large set of sub-algorithms or any intermediate representation. The cost of the representation, obviously, depends on the number of algorithms that compose the model. The accuracy of the complexity representation depends on the number and type of algorithms, in other words on their relevance in terms of complexity. A simple example of a complexity model is reported in figure 3 which reports a flow graph for the modeling of a processing composed by three algorithms f and associated occurrence complexity measures c. In this model, the processing is represented by tree algorithms f and by the associated counters c. Figure 4 reports a flow graph of the model of Figure 3 for which the complexity modeling of algorithms f1 and f3 have been designed by partitioning f1 and f3 in two new algorithms and their associated complexity models, the model is more accurate since the algorithms 1 and 3 are partitioned into other algorithms associated to new algorithm occurrence measures. It must be noted that what are considered as coding modes associated to specific coding options in the usual coding language, are from the point of view of the invention distinct algorithms, since different complexities correspond to different coding modes. It must be also noted that the executions of the same algorithm using different parameters can be considered as the executions of distinct algorithms associated to the corresponding parameters.

[0042] The mapping of the complexity into any software/hardware implementation is straightforward. By knowing the complexity (execution time) of each algorithm on a specific platform and by using the information of the actual occurrence measure of each algorithms in the decoding process, one can easily calculate the total execution time (what is denoted as complexity) for the processing of an incoming bit-stream.

[0043] The key issue of this approach is to be able to find the representation in terms of minimal set of algorithms and sub-algorithms (tools) that being representative of the processing complexity, yields the best approximation of the execution time. This is particularly evident for very complex processing task such as video decoding. The importance of a minimal set is to keep as compact as possible the cost of the complexity representation and, therefore, make possible its separate transmission at a low cost.

[0044] Another feature of the proposed model is that it can easily take into account parallel software or hardware implementations. No hypothesis has been done on the internal implementation of each algorithms. In fact, they can be implemented at any level of hardware or software parallelism. The case for which different algorithms of the model have parallel implementations can also be treated by simplifying the model and just ignoring some counters associated to algorithms implemented in parallel. Simple practical examples of these cases are given below after the complete definition of the modeling-prediction complexity problem.

6.2 A technique to predict video decoding complexity : Decoding Complexity Prediction by Coding Statistics (DCPCS)

[0045] Let us consider a specific software/hardware implementation of a video decoder, running on a specific hardware platform. Since no hypothesis is made on the architecture of the software or on the architecture of the hardware, the approach doesn't loose any generality.

[0046] Let us consider that the true complexity C* (execution time) for the decoding of a given input data (i.e. a single frame or a MPEG-4 VOP) is a function P*, specific for the software/hardware implementation at hand, of the sequence of all the various processing operations p_l 1=1,...,L needed to decode the incoming data :

$$C^* = P^*(p_1, p_2, .., p_l, ... p_L). \quad (1)$$

[0047] The p_l describe the sequence of operations that define the decoding process. They are, for instance, the parsing of the bit-stream, the sequence of operations for decoding the in various form Intra, Inter, Not-Coded, Shape, Padding, the inverse quantisation, the IDCT, the motion compensation in its various form OBMC, simple prediction, 4 vector prediction , the parameter that characterize the processing of each prediction mode (integer pel vector prediction, half pel vector prediction), the number of bitwise parsing operations, the number of non-zero DCT coefficients and so on.

Aylo Exhibit 1002
Page 265 of 1405

[0048] The actual expression of the function P* is certainly extremely difficult to obtain. On the other hand, the objective of the present invention is to avoid the necessity to know all the parameters that define the complexity, but just to be able to predict the true complexity C* with a good degree of approximation. The cost of the description of the exact complexity function P* in terms of all coding parameters would be, indeed, prohibitive for the purposes of the present invention.

[0049] Therefore, let us leave the problem of finding the exact expression of the complexity function P* and try to find a sufficiently accurate approximation F of the function P* which provides a prediction C, using the model proposed in the previous section. One is therefore limited to consider a number I of algorithms fi, with i=1,...,I with I << L, and the associated model counters c_i :

$$C = F(c_1 \cdot f_1, c_2 \cdot f_2, .. c_i \cdot f_i, .. c_1 \cdot f_1) = c_1 \cdot w_1 + c_2 \cdot w_2 + ... + c_i \cdot w_i + ... + c_1 \cdot w_1 \approx C^* \quad (2)$$

where the coefficients w_i related to each c_i are complexity weight corresponding to the model algorithm f_i which accounts for the specific software and hardware architecture at hand.

[0050] Let us suppose now, that the values of the vector c for the incoming data (frame or VOP) are known. If the vector w of the w_i weights is known, one can predict the complexity of the incoming VOP bit-stream by a simple linear combination.

6.3 A method to estimate the algorithms weight coefficients w

[0051] The values of the vector w can be obtained by direct measurements on the platform, if the complexity values (execution time) of each function are accessible. In general, they can be also indirectly estimated by means of a simple linear regression applied on a set of measures of the true complexities C^*_k, with k =1 ... K, obtained by K samples of data processing (i.e. K frames or K VOPs taken from different sequences) on the selected platform :

$$\begin{cases} C^*_1 = W_1 \cdot C_{11} + W_2 \cdot C_{12} + ... + W_i \cdot C_{1i} + ... + W_I \cdot C_{1I} \\ C^*_2 = W_1 \cdot C_{21} + W_2 \cdot C_{22} + ... + W_i \cdot C_{2i} + ... + W_I \cdot C_{2I} \\ ..... \\ C^*_k = ... \\ ..... \\ C^*_K = W_1 \cdot C_{K1} + W_2 \cdot C_{K2} + ... + W_i \cdot C_{Ki} + ... + W_I \cdot C_{KI} \end{cases} \quad (3)$$

[0052] In matrix form the estimated vector of weights w can be expressed by the classical least square minimization expression :

$$\hat{w} = (c^T c)^{-1} c^T C, \quad (4)$$

where T indicates the transpose matrix,
and -1 indicates the inverse matrix.

[0053] Obviously, the linear regression and the measures of the C_k complexities can be made off-line since they have as objective the estimation of the vector w which is characteristic of a specific software/hardware decoder implementation and do not depend on the incoming data.

[0054] Once the vector of the w parameters is estimated, the evaluation of the actual complexity C of the incoming bit-stream can be simply done by calculating the linear combination of the complexity function. In practice, this can be done by reading the actual c_i counter values written in an appropriate form in the incoming data, and calculating the value of F(c). This fact implies, for instance, that video compression standards should be able to insert complexity information in appropriate headers of frames or VOPs, or transmit them as accompanying information. Figure 7 reports the measured decoding time and Decoding Time Prediction (expressed in msec) versus the sequence VOP number of the texture decoding of MPEG-4 VM6 for the sequence Akyio (Background and Akyio VOPs) at different bit-rates. The decoder is the Momusys VM6 on a Sun Ultrasparc. The absolute value of the Decoding Time Prediction error is

Aylo Exhibit 1002
Page 266 of 1405

reported in the bottom curve near the 0 value.

6.4 Definition of the complexity models for the texture decoding of MPEG-4 VM6 VOPs

[0055] In the previous section, a model for decoding complexity has been developed, aiming at being at the same time related only to the processing and apt to be mapped on any platform implementation. These ideal requirements have been expressed in six points. A last requirement needs to be verified. Does the model and the associated complexity representation provide complexity (time) predictions sufficiently compact but accurate enough for the aims of the present invention? There is described in this paragraph an example of application of the modeling to the texture decoding of MPEG-4 VM6 VOPsm, excluding the processing of shape coding and the padding of macroblocks. This exclusion is just motivated by the fact that at the time of the modeling and simulations, one have investigated only the stable coding modules of the MPEG-4 VM. The same modeling principles can be applied to those and to all new MPEG-4 coding algorithms and modes.

[0056] Let us partition the decoding of each MPEG-4 VOP in the following seven algorithms which represent all the texture coding process of any VOP including luminance and chrominance (Intra, Inter, Skip, Advanced Prediction Mode (ADM), Normal Prediction Mode (NPM), DCT coefficient processing (DCT_coeff), and DCT line processing (DCT_line)). Let us associate to each coding-decoding algorithm a statistic measure defined as follows. For Intra, Inter and Skip, it counts each time a macroblock is coded by using the Intra, Inter or Non-Coded coding mode. For ADM and NPM the statistic measure indicates the number of blocks 8x8 predicted using advanced prediction modes (OBMC or four vectors for macroblock) or normal prediction modes. For DCT_coeff, the measure counts the total number of DCT coefficients per VOP, and DCT_line the total number of executed DCT 8x1 per VOP. All these counters can be easily inserted in any MPEG-4 compliant or any other video/audio encoder. Figure 13 reports a schematic diagram of the process of counting coding algorithms/modes/parameters and provides a statistic description of the decoding process according to any complexity model. This diagram represents the generation of the Complexity Information Header at the encoder side. The encoder increments the counters of the coding algorithms/modes/parameters that are used to encode the video frame or video object to generate the associated statistics signal (known as complexity information). At the end of the encoding of each video frame or video object the statistics of the used coding algorithms/modes/parameters is inserted in the Complexity Information Header. Figure 1 reports a simplified flow graph of the MPEG-4 VM6 texture decoding algorithms. Note that, in the proposed technique, it is not necessary to explicitly model the actual decoder flow graph. The above graph represents the schematic flow diagram of a possible simple VOP texture decoding model of MPEG-4. It has to be noticed that by estimating the complexity weights using the total VOP decoding measures as explained in the previous paragraph, it is not even necessary to explicitly specify all transitions in the decoding flow diagram model.

[0057] More accurate complexity models could be designed by introducing new algorithms and the associated measures of occurrences. An example of a more accurate model including the partitioning of APM and NPM into predicted or bi-directional modes and into half-pel motion compensation modes is reported in Figure 6 which represents an example of extension of the MPEG-4 texture decoding complexity model including the different mode prediction and half-pel motion compensation functions.

[0058] A reduction to simpler decoding models when many counters of different algorithms are available is straightforward. Some counters can be ignored if, in a specific software/hardware implementation, they do not implies any or any relevant complexity difference. For instance, if the implementation does not yields different processing time if a macroblock is predicted by a full pel or half pel vector, or if there is no difference between APM or NPM, all these counters can be ignored or can be combined. The specific action depends on the architecture and on the complexity features of the software/hardware decoding platform used.

[0059] As reported in these model examples, a meaningful partitioning of the algorithms in terms of complexity can be easily done only referring to the processing of the data and without the need of specifying any software or hardware implementation. Only in the phase of estimating the weights w_i and to perform the actual prediction it is necessary to refer to a specific hardware/software implementation.

[0060] The two simple models presented here are also capable of taking into account parallel implementations which are implemented in each algorithm such as any packed arithmetic implementation or higher levels of parallelism. Parallel implementations at levels of macroblock decoding are not likely to occur in MPEG-4 systems. Decoders need to respect a given sequential order in decoding the macroblocks that is intrinsic in the syntax of the video data flow. Moreover, for multimedia processor implementations, the ones that are particularly considered in this description, parallelism is likely to be implemented at much lower levels than macroblock decoding. Higher levels are indeed very unlikely to appear.

[0061] Platforms that include also some hardware module accelerators, can be taken into account for the model. If one have, for instance, the hardware implementation of a 8x8 DCT module, one can simply discard the DCT_coeff and DCT_line counters and simply use the combination of Intra and Inter counters to predict the DCT decoding complexity.

Aylo Exhibit 1002
Page 267 of 1405

Similar measures can be taken for other hardware accelerator modules.

[0062] The present complexity modeling approach can be applied to the audio/video decoding, composition and rendering of media objects or to any real-time processing of data.

7. EXPERIMENTAL RESULTS OF VIDEO DECODING COMPLEXITY PREDICTION

[0063] This section reports simulation results of the prediction of VOP decoding complexity for the MPEG-4 VM6 compression standard. Counters corresponding to the algorithms defined by the model of Figure 5 have been inserted in the Momusys VM6 encoder implementation to generate the mentioned statistic measures. These integer figures have been converted into percentage values referred to the actual number of VOP macroblocks, normalized to 8 bit values and have been supposed to be transmitted as a VOP complexity header for each VOP. The cost of this information transmission results of a few tenths of bits and can be considered negligible if compared to the total video bit-rate. Obviously, any other representation of the statistical coding information which has a sufficient precision can be employed. A schematic diagram of this decoding time (complexity) prediction signal generation process is reported in figure 14 which represents a schematic diagram of VOP video decompression interactions between the Video Decoder and the Video Decoder Supervisor when the Allocated Decoding Time (T_{Alloc})is lower than the Decoding Time Prediction (T_{Pred}). The Target Decoding Time and Decoder Time Reference are used to verify that the Decompressed Video elements are made available at the desired time. The Target Decoding Time is derived implicitly by the frame or sampling rate or by the information associated to each VOP or video frame (associated time stamps).

[0064] Measures of the texture decoding time of each VOP for different video sequence have been performed on a Sun UltraSparc platform. The decoder is the Momusys VM6 decoder.

[0065] The estimation of the weight w vector has been obtained by a linear regression applied on the measured decoding time of each VOP using the expression of equation (4). To demonstrate the robustness of the proposed technique, the complexity prediction results have been generated using a vector w, estimated using the complexity measurements of all the test sequences, except the one for which the prediction is calculated. For instance, to predict a sequence A, the weights estimated on sequences B and C have been used, and so on respectively for the other test sequences. Table 1 reports the results of the different estimates of the vector w for the sequences Akyio, Coastguard and Bream. As can be easily noticed, complexity (time) estimates from different sequences do not differ relevantly from one to the other. Disposing of a relatively large set of sequences with the associated measurements, one can easily perform off-line estimates of decoder weights w characteristic of the decoder at hand.

[0066] Figure 7 reports the measured VOP decoding time, the VOP decoding time prediction and the time prediction errors (in absolute value) for the two VOPs of the sequence Akyio. Figure 8 represents the measured decoding time and Decoding Time Prediction (expressed in msec) versus the sequence VOP number of the MPEG-4 VM6 for the sequence Bream (Background and Fish VOP) at different bit-rates. The decoder is the Momusys VM6 on a Sun Ultrasparc. The absolute value of the Decoding Time Prediction error is reported in the bottom curve near the 0 value. The results have been obtained at different bit-rates and all measures are expressed in msec. The same experiment results are reported for the sequences Bream and Coastguard. In figures 9, 10 and 11, as can be easily noticed, while the measured actual decoding time has strong variations for each VOP, the decoding time prediction is very accurate and the two curves for most of the samples are difficult to distinguish one from the other. In these experiments, the error of the prediction is lower than 5% for each VOP and in average is around 1-2%.

Aylo Exhibit 1002
Page 268 of 1405

W (msec/f)	Akiyo	Bream	Coastguard
Intra	38.68	38.84	38.15
Inter	30.95	29.60	29.56
Skip	5.55	5.53	5.51
DCT_line	2.15	0.37	1.24
DCT_coeff	1.47	6.58	6.18
APM	226.47	230.23	228.70
NPM	36.30	37.60	36.01

Table 1. Estimated complexity weights vector w for the texture decoding complexity model of Figure 5 for different sets of measures. The complexity relevance for each weight is given by the product of the algorithm count and by the weight itself.

[0067] The excellent results demonstrate that a relatively simple complexity model is able to predict the decoding time (complexity) with a very good degree of approximation, when appropriate coding statistics are available. The degree of approximation resulting from the proposed method, indeed, is more than adequate to enable an intelligent and efficient resource allocation strategy.

[0068] The description and discussion of the new possible schemes of interaction between the real-time OS and the video decoder are the subject of next sections.

[0069] Another important conclusion is the fact that a very complex process such as MPEG-4 texture video decoding can be described by a suitable high level coding statistic information which is not directly related to any software or hardware implementation. The vector of coding mode counter values is, in fact, a compact coding parameter and coding mode information of the compressed bit-stream. Moreover, the obtained results reports that it is not necessary to define a very detailed model of the software/hardware decoder platform. Very complex decoder features such as cash strategy or memory accesses which could have a non-strictly deterministic behavior, can be taken into account in a statistical sense when one estimate the value of the weight vector characteristic of the decoder platform used.

8. NEW SCHEMES OF DECODER-OS SCHEDULER INTERACTIONS

[0070] New schemes of interaction between the decoder and the OS scheduler of the multimedia processor are possible when decoding time prediction is feasible. These schemes of interaction can be designed in order to provide the best resource allocation policy able to guarantee real-time performances.

[0071] In Figure 12, an example of decoding scheme employing decoding time prediction is reported. This figure represents a decoding scheme example in which the Complexity Information of each VOP is processed by the Complexity Prediction Unit which provides the Decoding Time Prediction signal to the decoder and to the real-time OS/scheduler that allocates the necessary resources (Allocated Decoding Time) to the decoder or decoding processes. The Decoding Time Prediction unit reads the Complexity Information Header carried in each VOP/frame bit-stream header or transmitted as side information in any format, produces the decoding time prediction for each frame/VOP, and provides this information to the real-time OS scheduler. In this way, the OS scheduler can allocate the necessary resources (i.e. Allocated Decoding Time) for the decoding of each or all the VOPs respecting the associated real-time constraints. In case of non real-time encoding, the complexity information can be compressed and transmitted before the real-time transmission and real-time decoding of the video/audio bit-streams. The usefulness of such scheme is evident considering that many VOPs of unknown complexity (decoding time) need to be decoded within an associated time interval

Aylo Exhibit 1002
Page 269 of 1405

T_c. This scheme is able to guarantee real-time performances if the available video processing resources T_{Alloc} (i.e. Allocated Decoding Time) are sufficient for the decoding of all N incoming VOPs within the given time interval. Formally, the following relation has to be satisfied :

$$T_c \geq T_{Alloc} \geq \sum_{n=1}^{N} T_n(t) \tag{5}$$

where T_n is the decoding time of the nth VOP, with n=1,..., N, where N being the VO and VOP index and t being the VOP sampling time.

[0072] This simple procedure scheme is applicable in two cases. The first is when all the VOPs belongs to the same sampling time t (implicitly defined by the sampling rate or also carried by time stamps associated to each VOP). In this case, there is no additional delay introduced by the decoder. The second case occurs when the VOPs can also belong to sequential sampling periods, i.e. M sampling intervals of the reference time t denoted as Dt (also referred as different time stamps). In this case, an additional delay has to be considered and an output buffer needs to be added to the decoder :

$$T_c \geq T_{Alloc} \geq \sum_{n=1}^{N} \sum_{m=1}^{M} T_n(t+m\Delta t) \tag{6}$$

[0073] This second scheme enables to smooth out computational peaks on a larger number of VOPs and on a larger time interval than a single image sampling period Dt. In other words, expression (6) might be satisfied even when expression (5) is not for some values of the sampling time t. Obviously, this result is obtained at the cost of an additional delay and an additional output buffer.

[0074] When the available processing resources (i.e. time) are not sufficient to decode all VOPs, this simple strategy cannot guarantee the respect of the real-time constraints. In such cases, other form of interactions and new decoding techniques are needed.

[0075] The main idea is to reduce the complexity of the decoding in order to satisfy relation (5) and (6) accepting some degradation to the resulting image quality. These techniques are called here Computational Graceful Degradation (CGD).

[0076] A schematic diagram of the construction of the complexity information header at the encoder side is reported in Figure 13. The encoder, during the encoding process, increments the registers (counters) corresponding to a set of encoding algorithms/modes/parameters used for the actual VOP or frame compression. At the end of the process, the counters contains a compact encoding/decoding statistical information that is written in the Complexity Information Header.

9. COMPUTATIONAL GRACEFUL DEGRADATION (CGD)

[0077] The main idea of CGD is to make available results of poorer quality when the results of the desirable quality cannot be obtained in real time. When the complexity of the task exceeds the available processing power, real-time services, possibly of degraded quality, are nevertheless provided respecting all real-time constraints. This approach makes the guarantee of the correct scheduling of tasks possible, even when the computational system load fluctuates and relation (5) and (6) cannot be satisfied.

[0078] The concept of graceful system degradation using imprecise results has been previously introduced in the field of real-time computing with applications to control system and automation. In these contexts, imprecise results are defined as intermediate results with the property that the more recent and intermediate result is the closer one that approximates the final result (i.e. the idea only works in monotonically improving computations). For more details, the reader can refer to W.A. Halang, A.D. Stoyenko, Constructing predictable real time systems, Kluwer Academic Publisher, 1991 ; K. Lin, S. Natarjan, J. Liu, Imprecise results : Utilizing partial computations in real-time systems, Proceedings of IEEE Real-Time System Symposium, December 1987 ; J. Liu, K. Lin, S. Natarjan, Scheduling real time periodic jobs using imprecise results, Proceedings of the IEEE Real-Time System Symposium, December 1987 incorporated herein by reference.

[0079] In these cases the problems to be solved are more concerned with fault tolerance, I/O overloads, stability and reactivity to unexpected inputs, and the corresponding techniques are not suitable for purposes of real-time decoding of compressed video/audio bit-streams.

[0080] In contrast, the aim of the present invention is to be able to provide the guarantee of real-time video decoding

Aylo Exhibit 1002
Page 270 of 1405

with possible degradations of the image quality, i.e. reductions of the peak signal to noise ratio (PSNR), without loosing parts of the image or parts of the audio content. Such alternative is certainly much more desirable than other straightforward options such as cutting image portions or simply skipping frames. These latter options, as can be easily understood, can have catastrophic consequences on prediction based coding systems. In H.263 systems, for instance, skipping the decoding of a single frame or a portion of it implies the skipping of all future incoming frames (or portion of them) before the next incoming Intra frame. The time interval of such skipping operation might cover many seconds, and the information content loss might be very relevant. Such abrupt interruptions, which are perceptually very annoying, might he caused not only by the simple fluctuation of the decoding complexity load as discussed in the previous sections, but also by the fluctuations of the load of other non-video/audio processing tasks which reduce the available processing resources (time). Moreover, such events are in general out of the control of the video/audio decoder system and cannot be easily forecasted. Considering these facts, the importance of techniques able to filter out complexity decoding peak loads or able to reduce the average decoding computation load without causing perceptually annoying consequences is evident. Skipping B-VOP could be, in some cases, a viable option for video accepting the introduction of motion judder in the sequence reproduction. Unfortunately, the presence of B-frames or B-VOP cannot be guaranteed in any bit-stream and just at the right time for which a reduction of the complexity is needed to respect the real time constraints. For audio applications, any skip of part of the content has perceptually a very annoying result.

[0081] In contrast with what could be commonly thought, also the classical multi-resolution coding features such as spatial or temporal scalabilities are not really useful for the aims of the present invention. In general, they are used to optimize the transmission bandwidth usage, enabling the compatibility of services at different bit-rates and spatial resolutions, but they must be embedded in the original bit-stream at the encoder side. If they are not embedded, the changes of resolution at the decoder size can results more computationally expansive than the simple decoding at the given resolution.

[0082] Another important reason for the introduction of CGD is given by the definition of profiles, levels and conformance tests for video compression standards based on software/processor platforms. For the MPEG-4 standard, it is expected to have a range of applications that will be much broader than the one of MPEG-1 and MPEG-2. Two possible scenarios can be identified. The first is the definition of a limited number of profiles and levels based on computational power. The second is the possibility of a continuum of decoders of different computational power. The first scenario faces very difficult tasks such as the definition of decoder conformance testing, and at the end it may result in a jungle of too many different levels and profiles.

[0083] The second scenario is completely open to platforms of different computational power. It requires pure functional decoder conformance testing and is able to provide the maximum quality of the service that is achievable by the capabilities of each platform.

[0084] The second scenario, although much more attractive than the first, poses new questions and problems have not yet been studied in video and audio coding. It is mandatory that all decoders can decode all MPEG-4 services but at different levels of quality. It means that a scalability-based on computational power is needed. This new type of scalability should enable a graceful degradation of the video quality when less than theoretically necessary computational power is available at the decoding side. In contrast with MPEG-2 scalabilities that are discrete, embedded in the bit-stream and designed for specific applications, this computational power based scalability should be continuous and implemented in all MPEG-4 bit-streams.

[0085] From these arguments, the usefulness of CGD and its important features and applications are clear. The last key point that has left CGD as a good idea but of difficult usage and implementation for video coding was that it was not clear how to implement it efficiently. Now that the present invention has developed a technique able to predict with a good degree of approximation the required decoding time of each decoding process, one have the key for useful and efficient implementation of CGD techniques.

9.1 Principles of decoding using CGD techniques

[0086] Let us consider a complexity model such as the one of Figure 3 composed by a set of I algorithms fi, with i = I,...,I and suppose to be able to design for all, or only for a subset S^{CGD} of the functions f_i composing the model, new functions f_i^{CGD} which have the following three properties :

1) the execution of a function f_i^{CGD} yields results that are an approximation of the ones produced by the original function f_i,
2) the execution of f_i^{CGD} does not prevent the exact execution of any subsequent decoding functions,
3) the weight w_i^{CGD} associated to f_i^{CGD} in the considered decoder implementation is lower than the corresponding w_i associated at f_i (i.e. $w_i^{CGD} < w_i$).

[0087] For a schematic representation of a complexity model which includes new degraded functions, one can refer

Aylo Exhibit 1002
Page 271 of 1405

to Figure 15 which represents a schematic diagram of a complexity model including CGD algorithms, CGD counters (b_i) and new state transitions generated by the new CGD functions. In this example $f_1, f_3 \hat{I} S^{CGD}$, and $f_2 \hat{I} S^{orig}$. In other words, for video decoding for instance, the execution of any f_i^{CGD} might produces PSNR degradations in the reconstruction of some image parts, but not prevents from correctly decoding the rest of the frame or VOP.

[0088] Let us partition the functions forming the complexity model into two disjoint sets, S^{Orig} and S^{CGD} respectively, for the ones that do not admit a CGD version and for the ones for which it is possible to design a CGD version.

[0089] Let us consider for simplicity an image sequence composed by a sequence of single frames or single VOPs (i.e. N=1) and suppose that expression (5) is not satisfied for some values of the decoder reference time t :

$$T(t) > T_{Alloc} \tag{7}$$

where T(t) is the decoding time of the VOP at hand at the sampling time t, and to define as processing deficit T_{Def} the time for which the decoding complexity exceeds the Allocated Decoding Time :

$$T(t) - T_{Alloc} = T_{Def} \tag{8}$$

[0090] Since the decoding time can be approximated by its prediction (Decoding Time Prediction) it can be expressed by :

$$T(t) \cong T_{Pred}(t) = \sum_{i=1}^{I} c_i w_i \tag{9}$$

where the c_i are the counters of the complexity model and the w_i are the corresponding complexity weights.

[0091] The lower limit of the decoding time executing all CGD algorithms f_i^{CGD} at the place of f_i for any $i \hat{I} S^{CGD}$, is given by :

$$T_{CGD} = \sum_{i \in S_{CGD}} c_i w_i^{CGD} + \sum_{i \in S_{Orig}} c_i w_i \tag{10}$$

[0092] If $T_{CGD} < T_{Alloc} < T_{Pred}$ it is therefore possible to guarantee the real-time constraints by executing the f_i^{CGD} algorithms and accepting approximate decoding results. The difference $T_{Pred} - T_{CGD}$ can also be expressed as :

$$T_{Pred} - T_{CGD} = \sum_{i \in S_{CGD}} c_i (w_i - w_i^{CGD}) \tag{11}$$

[0093] If $T_{CGD} < T_{Alloc} < T_{Pred}$, it is therefore possible to find for each c_i with $i \hat{I} S^{CGD}$, a couple of integer coefficients a_i, b_i, for which :

$$a_i + b_i = c_i \tag{12}$$

and denoting as $T_{Process}$ any decoding time interval which respects the real time constraints (obtained by the implicit sampling rate or expressed using appropriate time stamps associated to each VOP or video frames, or audio packet), one can express the desired process decoding time as any decoding time value that satisfy the expression :

$$T_{process} = \sum_{i \in S_{CGD}} (a_i w_i^{CGD} + b_i w_i) + \sum_{i \in S_{Orig}} c_i w_i \leq T_{Alloc} \tag{13}$$

Aylo Exhibit 1002
Page 272 of 1405

[0094] Obviously for any Allocated Decoding Time the aim is to respect the real-time constraints with the lowest degradation in the video/audio content. Associating a degradation measure to each w_i^{CGD} it is possible to minimize the degradation for a given TAlloc. Therefore, the aim is to find the set $S_{a,b}$ of couple of coefficients a_i, b_i, with i Î S^{CGD}, that satisfy expression (12) and minimize the video or audio degradations expressed as difference with the non degraded output :

$$S_{a,b}=\text{argmin}(a,b)_{i\in S_{CGD}}:(PSNR\text{-}PSNR(a_i,b_i)) \tag{14}$$

[0095] The determination of the set $S_{a,b}$ can be done by means of any suitable optimization method or any suitable optimization heuristic by the Video Decoder Supervisor, schematically represented in Figure 14. In summary, the Video Decoder Supervisor receives from the OS the Allocated Decoding Time signal (T_{Alloc}), the decoding time prediction signal (TPred) from the Complexity Prediction unit, and the Complexity Information signal represented by the actual values of the coding/decoding statistic (set of ci coefficients) for the VO. Since the Video Decoder Supervisor knows the values of the set of w_i and w_i^{CGD} it can determine, by a suitable standard optimization technique, the corresponding set of coefficients a_i,b_i such that expression (12) is satisfied. Having defined the set of a_i,b_i coefficients the CGD decoding is partially defined. In other words, the set of coefficients defines for each i Î S^{CGD} how many degraded algorithms (f_j^{CGD}) at the place of original algorithms (f_j) need to be executed.

[0096] Since the determination of the degradation coefficients is done on the basis of the decoding time prediction, the video decoder can generate a Decoding Status signal reporting the status of the decoding (i.e. the fraction of the bit-stream or the fraction of the VO already decoded). The signal representing the Decoding Status of the decoding process reports to the Video Decoder Supervisor an information which can be used to correct the Computational Graceful Degradation Directive signal (i.e. the setting of the a_j,b_j coefficients in this example) if the Current Decoding Time and the Current Decoding Time Prediction signal are not conformant. Current Predicted Decoding Time means the Decoding Time Prediction calculated by using elation (13), only counting the occurrences of the decoding algorithms actually used to decode the fraction of the decoded bit-stream at the current reference time (time for which the decoding status signal is generated). The term not conformant means that the actual Decoding Time is greater than the Decoding Time Prediction of an amount of time which overcomes statistical variations which are functions of the complexity information, thus reporting possible errors in the Decoding Time Prediction signal.

[0097] The described interactions between OS, the Decoder and Video Decoder Supervisor can be easily extended for the case of sequences containing multiple VO at the same sampling time (associated time stamps), or belonging to a defined time interval containing several sampling periods (several associated time stamps). Expressions (10,11,13,14) can be easily modified including the appropriate summation over all VOP. Therefore, in the case of N VOP at the same sampling time t, one have :

$$T_N(t)=\sum_{n=1}^{N} T_n(t)>T_{Alloc} \tag{15}$$

where T_N (t) is the total decoding time of the N VOP.

$$T_N(t)\cong T_{Pred,N}=\sum_{n=1}^{N} T_{Predn}(t)=\sum_{n=1}^{N}\sum_{i=1}^{I} c_{i,n}w_i \tag{16}$$

$$T_{CGD,N}=\sum_{n=1}^{N}\left(\sum_{i\in S_{CGD}} c_{i,n}w_i^{CGD}+\sum_{i\in S_{Orig}} c_{i,n}w_i\right) \tag{17}$$

$$T_{Pred,N}\text{-}T_{CGD,N}=\sum_{n=1}^{N}\sum_{i\in S_{CGD}} c_{i,n}(w_i\text{-}w_i^{CGD}) \tag{18}$$

$$a_{i,n}+b_{i,n}=c_{i,n} \tag{19}$$

Aylo Exhibit 1002
Page 273 of 1405

$$T_{Process,N}=\sum_{n=1}^{N}\Big(\sum_{i\in S_{CGD}}(a_{i,n}w_i^{CGD}+b_{i,n}w_i)+\sum_{i\in S_{Orig}}c_{i,n}w_i\Big)\le T_{Alloc} \quad (20)$$

$$S_{a,b,N}=\text{argmin}_{(a,b)_i\in S_{CGD}}\Big(\sum_{n=1}^{N}(PSNR-PSNR(a_{i,n},b_{i,n}))\Big) \quad (21)$$

or :

$$S_{a,b,N}=\text{argmin}_{(a,b)_i\in S_{CGD}}\Big(\sum_{n=1}^{N}\alpha_n(PSNR-PSNR(a_{i,n},b_{i,n}))\Big) \quad (22)$$

where a_n is a semantic degradation coefficient which might be available in the coded bitstreams.

[0098] For N Video Objects and M sampling time intervals Dt (denoted also as time stamps) one have :

$$T_{N,M}=\sum_{m=1}^{M}T_N(t+m\Delta t)=\sum_{m=1}^{M}\sum_{n=1}^{N}T_n(t+m\Delta t)>T_{Alloc} \quad (23)$$

$$T_{N,M}\cong T_{Pred,N,M}=\sum_{m=1}^{M}\sum_{n=1}^{N}T_{Predn}(t+m\Delta t)=\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{i=1}^{I}c_{i,n,m}w_i \quad (24)$$

$$T_{CGD,N,M}=\sum_{m=1}^{M}\sum_{n=1}^{N}\Big(\sum_{i\in S_{CGD}}c_{i,n,m}w_i^{CGD}+\sum_{i\in S_{Orig}}c_{i,n,m}w_i\Big) \quad (25)$$

$$T_{Pred,N,M}-T_{CGD,N,M}=\sum_{m=1}^{M}\sum_{n=1}^{N}\sum_{i\in S_{CGD}}c_{i,n,m}(w_i-w_i^{CGD}) \quad (26)$$

$$a_{i,n,m}+b_{i,n,m}=c_{i,n,m} \quad (27)$$

$$T_{Process,N,M}=\sum_{m=1}^{M}\sum_{n=1}^{N}\Big(\sum_{i\in S_{CGD}}(a_{i,n,m}w_i^{CGD}+b_{i,n,m}w_i)+\sum_{i\in S_{Orig}}c_{i,n,m}w_i\Big)\le T_{Alloc} \quad (28)$$

Aylo Exhibit 1002
Page 274 of 1405

$$S_{a,b,N,M} = \text{argmin}_{(a,b)_{i \in S_{CGD}}} \left(\sum_{m=1}^{M} \sum_{n=1}^{N} (PSNR - PSNR(a_{i,n,m}, b_{i,n,m})) \right) \quad (29)$$

or

$$S_{a,b,N,M} = \text{argmin}_{(a,b)_{i \in S_{CGD}}} \left(\sum_{m=1}^{M} \sum_{n=1}^{N} \alpha_n (PSNR - PSNR(a_{i,n,m}, b_{i,n,m})) \right) \quad (30)$$

9.2 Extension of the techniques to video/audio composition

[0099] The methods aiming at predicting the decoding time (i.e. generating a Decoding Time prediction Signal) of video frames or video objects, so as to respect the real-time constraints specified implicitly by the audio/video rates or by the timing information carried by the associated time stamps, can be easily extended to the video/audio composition processes. Composition process means the action of putting decompressed video or audio objects together in the same representation space. In a similar way of the decoding process, one can generate a Composition Time Prediction for each composition process according to the interactions described in Figure 12. Such Composition Time Prediction is a function of the occurrence of composition algorithms, composition modes and composition parameters (associated composition statistics). Differently from the decoding process, composition algorithms, modes and parameters can be also provided by an interactive user interface. In a similar way the Composition Allocated Time can be generated as function of the Composition Time Prediction. The interactions between the compositor and the OS can be described as in Figure 12 where the Decoder is substituted by the Compositor.

[0100] The technique of associating Decoding Degradation Algorithms and decoding by using degradation algorithms described in section 6.1 and Figure 14 can be applied by substituting respectively the Allocated Decoding Time by the Decoding Time Prediction, the Target Decoding Time and the Decoder Reference Time by the Allocated Composition Time, the Composition Time Prediction by the Target Composition Time and the Compositor Reference Time. All relations and equations reported in section 6.1 referring to the decoding process can be referred to the composition process.

9.3 Examples of possible degradation modes

[0101] The decoding operations of a video compression algorithm can be divided into two main categories :

1) operations that cannot be simplified since they may prevent the extraction and usage of relevant coding information,
2) operations that can be simplified, possibly leading to imprecise or degraded video results.

[0102] To the first category typically belong all bit-stream parsing operations, entropy decoding (i.e. VLD and arithmetic coding), and all operations related to synchronization and to the syntax of the decoding process. For these operations, no degradation mode definition is possible, otherwise it would imply skipping image portions or entire frames since either the data or the coding modes and coding parameters are lost.

[0103] To the second category, for instance, belong operations such as the IDCT, the prediction with and without motion compensation in its various forms (PB/B-frame prediction, overlapped block motion compensation (OBMC), half pixel interpolation and so on, padding, shape coding. Simplifications of such operations applied to pixels or transform coefficients are always possible but at the expenses of imprecise results. The challenge can be expressed as follows : given a specific compression standard, software and processor architecture, implement it and find the f_i^{CGD} which provides the maximum complexity reduction ($w_i - w_i^{CGD}$) versus the minimum image/audio quality degradation (usually measured in PSNR). Many are the algorithms that can be simplified. We report here only two examples that illustrate the idea.

9.3.1 IDCT simplified modes

[0104] IDCT on a 8x8 block can be decomposed into 16 IDCT operation on a 8x1 block, or on IDCT considering only

Aylo Exhibit 1002
Page 275 of 1405

the lowest 4x4 or 2x2 coefficients. Skipping higher order 8x1 IDCT or enabling processing only 2x2 or 4x4 coefficients constitute some alternatives which are less complex than the standard 8x8 IDCT. Obviously, these less demanding alternatives yield a filtering of the high frequency possibly contained in the higher order coefficients.

9.3.2 Simplified prediction modes

[0105] Many prediction modes can be simplified at the expenses of some PSNR degradations. In particular, the most complex such as the OBMC with or without half-pel motion compensation can yield relevant complexity decreases of both computational complexity and memory bandwidth by implementing simpler algorithms.

10. EXAMPLES OF RESULTS FOR THE MPEG-4 VIDEO COMPRESSION STANDARD

[0106] This section reports simulation results of the decoding of VOP using the complexity model and decoding time prediction results described in Section 4. The MPEG-4 VM6 algorithm in the Momusys VM6 implementation modified adding degradation functions to the advanced prediction modes has been used for the simulations. A fixed amount of processing time is allocated for each VOP composing the sequences. In case the decoding time prediction (T_{Pred}) for each VOP exceeds the fixed allocated time (T_{Alloc}), an appropriate number of advanced prediction modes are executed using degraded functions in order to satisfy relation (13).

[0107] Figures 16, 17, 18 report the decoding results at 64, 128 and 256 kb/s for the four VO (Small Boat, Big Boat, background and Water) of the sequence Coastguard. On the left, the diagrams report the decoding time of each VO expressed in msec versus the VOP number. Two curves are reported : original decoding and decoding using degradation function in order to satisfy the decoding time limits. The PSNR image degradations corresponding to the execution of degraded functions are reported on the right.

[0108] Figure 16 reports on the left standard decoding time and CGD decoding time (in msec) for the four VO of the sequence Coastguard at 64 kb/s. Each VO has a upper bound for the CGD decoding time. On the right the PSNR loss of the CGD decoding with reference to the standard decoding.

[0109] Figure 17 reports on the left standard decoding time and CGD decoding time (in msec) for the four VO of the sequence Coastguard at 128 kb/s. Each VOP has a upper bound for the CGD decoding time. On the right the PSNR loss of the CGD decoding with reference to the standard decoding.

[0110] Figure 18 reports on the left standard decoding time and CGD decoding time (in msec) for the four VO of the sequence Coastguard at 256 kb/s. Each VO has a upper bound for the CGD decoding time. On the right the PSNR loss of the CGD decoding with reference to the standard decoding.

[0111] The implementation of CGD base on the decoding time prediction generation technique enables, within the discussed limits, the a priori setting of the decoding time guaranteeing the possibility of a scheduling of each decoding task that satisfy real time constraints, even if the available resources (Allocated Decoding Time) are inferior to the theoretically necessary ones. Moreover, the degradations introduced are visually acceptable and they can be minimized according to expression (14) by suitable techniques. The results obtained with a PSNR degradated are certainly much more desirable than introducing sudden interruptions and image content losses to the video sequence, and enables an efficient resource allocation policy.

Claims

1. Method for primary processing a digital signal comprising the steps of :

   - providing a digital signal to a primary processing unit ;
   - primary processing said signal with said primary processing unit according to a plurality of primary processing algorithms to provide primary processed output signals,
   - determining statistics of use of at least one of said primary processing algorithms,
   - providing a digital statistics signal representative of said statistics for each of said primary processed output signals
   - associating to each of said primary processed output signals its own statistics signal ;
   - determining a complexity primary processing information signal based on said statistic signal.

2. Digital primary processed output signal, generated by a primary processing unit using primary processing algorithms and being intended to be transmitted to a receiving device wherein said primary processed output signal comprises data bit streams, each of said data bit streams comprising a main bit string representative of said primary processed output signal and a secondary bit string representative of the statistics of use of said primary processing algorithms associated to said main bit string.

Aylo Exhibit 1002
Page 276 of 1405

3. Method for secondary processing a secondary digital output signal according to claim 2 comprising the steps of :

(a) reading said secondary bit string ;
(b) determining, from the secondary bit string, primary processing algorithms to be used to secondary process said primary output signal associated to said main bit string, for generating a complexity primary processed information signal for each signal ;
(c) allocating a secondary processing time and/or operating system resources as a function of said complexity primary processed information signal for secondary processing of said primary output signal ;
(d) sending main bit string to secondary processing means ;
(e) secondary processing said main bit string using said allocated secondary processing time and/or operating system resources, according to a secondary process and
(f) generating a secondary processed signal.

4. Method for coding a digital signal comprising the steps of :

- providing an uncompressed digital signal including audio packets, video frames, video planes, or audio/video objects to an encoder ;
- compressing said signal with said encoder according to a compression syntax, said encoder using a plurality of coding algorithms to provide a compressed signal including coded audio packets, video frames, video planes or audio/video objects,
- determining statistics of use of at least one of decoding algorithms necessary to decode each of said coded audio packets, video frames, video planes or audio/video objects from said associated coding algorithms, and/or determining statistic of use of at least one of coding algorithms,
- providing a digital statistics signal representative of said statistics for each of said coded audio packets, video frames, video planes or audio/video objects,
- associating to each of said coded audio packets, video frames, video planes or audio/video objects its own statistics signal.

5. Coding method according to claim 3, wherein the step of determining statistics consists in determining the statistics of use of some of decoding algorithms, wherein to each decoding algorithm is associated at least one decoding mode and/or decoding parameter and wherein the statistics signal is based on the decoding modes and/or decoding parameters.

6. Coding method according to claim 3 or 4, wherein the step of determining statistics further consists in determining the statistics of use of some of coding algorithms when said coding algorithms and the compression syntax allow the determination of corresponding decoding algorithms necessary to decode each of said coded audio packets, video frames, video planes or audio/video objects wherein to each coding algorithm is associated at least one coding mode and/or coding parameter and wherein the statistics signal is based on the coding modes and/or coding parameters necessary to decode when using said coding algorithms.

7. Coding method according to claim 4, wherein the statistic signal is generated by juxtaposing digital words each representing the occurrences of use of a decoding mode and/or decoding parameter necessary to decode said associated audio packet, video frame, video plane or audio/video object.

8. Coding method according to claim 5 and/or 6, wherein the statistic signal is generated by juxtaposing digital words each representing the occurrences of use of a coding mode and/or coding parameter necessary to decode said associated audio packet, video frame, video plane or audio/video object.

9. Coding method according to any of claims 3 to 7, wherein the step of associating consists in inserting said statistics signal into said compressed signal for each associated coded audio packet, video frame, video plane or audio/video object.

10. Coding method according to any of claims 3 to 8, wherein the step of determining statistics consists in computing the occurrence of use of said coding and/or decoding algorithms for determining a complexity information for each coded audio packet, video frame, video plane or audio/video object.

11. Digital compressed signal including coded audio packets, video frames, video planes or audio/video objects, said signal being generated by an encoder using a compression syntax and coding algorithms and being intended to be

Aylo Exhibit 1002
Page 277 of 1405

transmitted to a receiving device wherein said signal is decoded according to decoding algorithms, and wherein said signal comprises at least a layer of coded audio packets, video frames, video planes or audio/video objects bit streams, each of said bit streams comprising a main bit string representative of the coded audio packets, video frames, video planes or audio/video objects and a secondary bit string representative of the statistics of use of said coding and/or decoding algorithms associated to said main bit string.

12. Digital compressed signal, according to claim 10, wherein said secondary bit string is inserted in said layer of said bit streams.

13. Digital compressed signal, according to claim 10, further comprising a plurality of parallel layers of coded audio packets, video frames, video planes, or audio/video objects bit streams, wherein said main bit string is inserted or transmitted in a layer different to said second bit string.

14. Digital compressed signal, according to claim 11 or 12, wherein said secondary bit string is inserted or transmitted prior to said main bit string.

15. Digital compressed signal, according to any of claims 10 to 13, wherein to each decoding algorithm is associated at least one decoding mode and/or decoding parameter and wherein said secondary bit string consists of words each representing the occurrences of use of the decoding modes and/or decoding parameters.

16. Digital compressed signal, according to claim 10 or 14, wherein to each coding algorithm is further associated at least one coding mode and/or coding parameter and wherein said secondary bit string consists of words each representing the occurrences of use of the coding modes and/or coding parameters, said coding algorithms and the compression syntax allowing the determination of corresponding decoding algorithms necessary to decode each of said coded audio packets, video frames, video planes or audio/video objects.

17. Method for decoding a digital compressed signal according to any of claims 10 to 15 comprising the steps of :

(a) reading said secondary bit string ;
(b) determining, from the secondary bit string, decoding algorithms to be used to decode said coded audio packets, video frames, video planes or audio/video objects, associated to said main bit string, for generating a complexity information signal for each coded audio packet, video frame, video plane or audio/video object ;
(c) allocating a decoding time and/or operating system resources as a function of said complexity information signal for the decoding process of said compressed audio packet, video frames, video planes or audio/video objects ;
(d) sending main bit string to decoding means ;
(e) decoding said main bit string using said allocated decoding time and/or operating system resources, according to a decoding process and
(f) generating a decompressed signal including decoded audio packets, video frames, video planes or audio/video objects.

18. Method for decoding a digital compressed signal according to claim 16 wherein the step determining decoding algorithms is also based on the occurrence of use of the decoding modes and/or decoding parameters necessary to decode the said audio packets, video frames, video planes or audio/video objects.

19. Method for decoding a digital compressed signal according to claim 16 or 17 wherein the step determining decoding algorithms is further based on the occurrence of use of the coding modes and/or coding parameters used by said coding algorithms, wherein said coding algorithms and the compression syntax allow the determination of corresponding decoding algorithms necessary to decode each of said coded audio packets, video frames, video planes or audio/video objects

20. Decoding method according to any of claims 16 to 18 further comprising the step of determining a decoding time prediction signal as a function of said complexity information signal ; said decoding time prediction signal being used for the step of allocating said decoding time and/or operating system resources for optimizing said decoding process and or decoding processes.

21. Decoding method according to claim 19 wherein the step of decoding the main bit string consists in

Aylo Exhibit 1002
Page 278 of 1405

- generating an allocated decoding time signal,
- sending said decoding time prediction signal and allocated decoding time signal to decoder supervisor means ;
- generating and sending a degradation directive signal to said decoding means, as a function of said decoding time prediction signal, allocated decoding time, so as to define the use of number and type of decoding algorithms and degradation algorithms to be used in the decoding process of said main bit string.

22. Decoding method according to claim 20 wherein the step of decoding the main bit string further consists in :

- generating a decoder time reference signal and a target decoding time signal,
- sending said decoder time reference signal and said target decoding time signal to decoder supervisor means ;
- generating and sending said degradation directive signal to said decoding means while also taking into account said decoder time reference signal and said target decoding time signal.

23. Decoding method according to claim 20 or 21 wherein the step of decoding the main bit string further consists in :

- generating a decoding status signal from said decoding means
- sending said decoding status signal to said decoder supervisor means from the said decoding means ;
- generating and sending said degradation directive signal to said decoding means while also taking into account said decoding status signal.

24. Decoding method according to claim 16 wherein steps (a) to (c) are carried out prior to step (e).

25. Decoding method according to any of claims 20 to 23, further comprising the step of sending said complexity information signal to said decoder supervisor means.

26. Decoding method according to any of claims 20 to 24 wherein the step of generating said degradation directive signal consists in associating at least one degradation algorithm to at least one decoding algorithm, said decoding algorithm and said degradation algorithm being associated to a complexity coefficient/weight representative of the execution time of said decoding algorithm and said degradation algorithm.

27. Decoding method according to claim 25 wherein said complexity coefficients/weights are dependent on the decoder used for implementing said decoding method.

28. Decoding method according to claim 25 or 26, wherein said decoding time prediction is a function of the occurrence of use of said decoding algorithm and said complexity coefficients/weights according to the following equation :

$$T(t) \cong T_{Pred}(t) = \sum_{i=1}^{I} c_i w_i \qquad (9),$$

where T is the time necessary to decode a coded audio packet, video frame, video plane or audio/video object, T_{Pred} is the decoding time prediction of said audio packets, video frames, video planes or audio/video objects, c_i is the occurrence of said decoding algorithm, and w_i is the complexity coefficient/weight.

29. Decoding method according to claim 27, wherein when the decoding time prediction is a function of the occurrence of use of said decoding algorithms and degradation algorithms, said degradation algorithms being associated to a complexity coefficient/weight.

30. Decoding method according to claim 27 or 28, wherein when the decoding time prediction is greater than the allocated decoding time, one uses a degradation method according to which one selects the degradation algorithm instead of its associated decoding algorithm according to equation :

$$T_{process} = \sum_{i \in S_{CGD}} (a_i w_i^{CGD} + b_i w_i) + \sum_{i \in S_{Orig}} c_i w_i \leq T_{Alloc} \qquad (13)$$

Aylo Exhibit 1002
Page 279 of 1405

where, $T_{Process}$ is the prediction time necessary to decode a audio packets, video frames, video planes or audio/video objects, said decoding time prediction taking account the degradation, c_i is the occurrence of decoding algorithm, a_i is the number the selected degradation algorithms, w_i^{CGD} is the complexity coefficient/weight of the degradation algorithm, b_i is the number of the non selected degradation algorithms, and w_i is the complexity coefficient/weight, where $a_i + b_i = c_i$, so as to reach a $T_{Process}$ lesser than or equal to said allocated decoding time.

31. 30. Decoding method according to claim 28 or 29, wherein said decoding status signal is also used for monitoring the current decoding time of a coded audio packets, video frames, video planes or audio/video objects and make sure that said current decoding time is in conformity with the associated decoding time prediction.

32. Decoding method according to claim 30, wherein if said current decoding time is not in conformity with the associated predicted decoding time, then said decoder supervisor means uses at least one further degradation algorithm defined in claim 29.

33. Decoding method according to claim 31, wherein if said current decoding time is not in conformity with the associated decoding time prediction, then said decoder supervisor means further uses said complexity information signal.

34. Decoding method according to any of claims 29 to 32 for a plurality of audio packets, video frames, or video planes, some of them belonging to different audio/video objects, wherein when the decoding time prediction is the sum of the decoding time prediction of each of said audio packets, video frames or video planes of said plurality, said decoding time prediction being defined in claim 28.

35. Coding system for an uncompressed digital signal including audio packets, video frames, video planes or audio/video objects, comprising :

- an audio/video encoder receiving said audio packets, video frames, video planes or audio/video objects on one input terminal, said encoder being intended to code said signal according to a compression syntax by using a plurality of coding algorithms, and to provide, on one output terminal, a compressed signal including coded audio packets, video frames, video planes or audio/video objects intended to be decompressed according to associated decoding algorithms ;
- means for determining statistic of use of at least one of said coding algorithms, and/or means for determining statistic of use of at least one of said decoding algorithms ;
- and means for generating a digital statistic signal representative of said statistics for each of said coded audio packets, video frames, video planes or audio/video objects.

36. Coding system according to claim 27, wherein said statistics determining means comprise computing means for counting the occurrence of use of coding modes and/or coding parameters used by the encoder for encoding said signal, when using said coding algorithms.

37. Coding system according to claim 34 or 35, wherein said statistics determining means further comprise computing means for counting the occurrence of use of decoding modes and/or decoding parameters from said coding algorithms used by the encoder for encoding said signal.

38. Coding system according to any of claim 34 to 36, further comprising means for generating a compressed output signal including coded audio packets, video frames, video planes or audio/video objects wherein each of said coded audio packet, video frame, video plane or audio/video object is associated with its statistics signal.

39. Decoding system for a digital signal including coded audio packets, video frames, video planes or audio/video objects according to any of claims 5 to 15, comprising :

- a decoder receiving said main bit string, representative of a coded audio packets, video frames, video planes or audio/video objects, on a first input terminal and providing, on an output terminal, a decompressed signal including decoded audio packets, video frames, video planes or audio/video objects,
- resource allocation means ;
- means for complexity prediction receiving, on one input terminal, said secondary bit string, and providing to resource allocation means a decoding time prediction signal which is determined as a function of the information contained in said secondary bit string, said resource allocation means providing to the decoder information representative of the allocated decoding time.

Aylo Exhibit 1002
Page 280 of 1405

40. Decoding system according to claim 38 wherein said resource allocation means comprises an operating system or a layer of an operating system.

41. Decoding system according to claim 38 or 39 wherein said decoding time prediction signal is also provided to a second input terminal of said decoder.

42. Decoding system according to any of claims 38 to 40, wherein

- said decoder comprises decoder supervisor means and audio/video decoding means,
- said decoder supervisor means receiving, on a first input terminal, said allocated decoding time signal and, on a second input terminal, said predicted decoding time signal and providing on one output terminal a degradation directive signal,
- said decoding means receiving on a first input terminal said main bit string and on a second input terminal said degradation directive signal and providing on an output terminal said decompressed signal including decoded audio packets, video frames, video planes or audio/video objects.

43. Decoding system according to claim 41 wherein said decoder further receive a complexity information signal on third input terminal.

44. Decoding system according to claim 41 wherein said decoder further receive a decoder time reference signal and a target decoding time signal respectively on fourth and fifth input.

45. Decoding system according to any of claims 41 at 43 wherein said decoding means are arranged to provide a decoding status signal to a sixth input of the decoder supervisor means.

46. Decoding system according to claim 44 wherein said complexity information signal is applied to said decoder supervisor means.

47. Method for composing a digital signal comprising the steps of :

- providing an uncompressed video element including video frames and/or video planes and/or video objects, textures, texts, 2 and 3 dimensional graphic primitives, 2 and 3 dimensional models to a composing device ;
- composing said video element with said composing device according to a composition syntax an/or a interactivity information, said composing device using a plurality of composing algorithms necessary to compose each of said video elements and provide a representation in a common visual representation space of the said multiple elements,
- determining statistics of use of at least one of said composing algorithms for generating a complexity information
- associating to each of said video elements its own complexity information.

48. Composition method according to claim 46 further consisting in :

- providing an uncompressed digital audio element including audio packets, audio objects, synthetic audio objects
- composing said audio element with said composing device according to a composition syntax, said audio composing device using a plurality of composing algorithms necessary to compose each of said audio elements and provide a representation in a common audio representation space
- determining statistics of use of at least one of said algorithms for generating a complexity information.
- associating to each of said audio element its own complexity information.

49. Composition method according to claim 46 or 47, wherein to each composition algorithm is associated at least one composition mode and/or composition parameter and wherein the statistics is based on the occurrence of use of composition modes and/or composition parameters.

50. Method for composing a digital signal according to any of claims 46 to 48 comprising the steps of :

(a) reading the complexity information associated to at least one of said video element and/or audio element
(c) allocating a composition time and/or operating system resources as a function of said complexity informa-

Aylo Exhibit 1002
Page 281 of 1405

tion signal for the composition process of said associated video element and/or audio element ;
(d) sending said video element and/or audio element to composition means ;
(e) compose in a common representation space said video element and/or audio element using said allocated composition time and/or operating system resources, according to a composition process, and
(f) generating a common representation space including said video element and/or audio element according to the composition syntax.

51. Method for composing digital video/audio elements according to claim 49 wherein the step determining composition algorithms is also based on the occurrence of use of the composition modes and/or composition parameters necessary to compose the said composition elements in a common composition space.

52. Composition method according to any of claims 48 to 50 further comprising the step of determining a composition time prediction signal as a function of said complexity information ; said composition time prediction signal being used for the step of allocating said composition time and/or operating system resources for optimizing said composition process and or composition processes.

53. Composition method according to claim 50 wherein the step of composing the audio/video elements consists in

- generating an allocated composition time signal,
- sending said composition time prediction signal and allocated composition time signal to composition supervisor means ;
- generating and sending a degradation directive signal to said composition means, as a function of said composition time prediction signal, allocated composition time, so as to define the use of number and type of composition algorithms and degradation algorithms to be used in the composition process of said video/audio composition elements.

54. Composition method according to claim 52 wherein the step of composing the video/audio elements further consists in :

- generating a compositor time reference signal and a target composition time signal,
- sending said compositor time reference signal and said target composition time signal to composition supervisor means ;
- generating and sending said degradation directive signal to said composition means while also taking into account said compositor time reference signal and said target composition time signal.

55. Composition method according to claim 52 or 53 wherein the step of composing the audio/video elements further consists in :

- generating a compositor status signal from said composition means
- sending said compositor status signal to said composition supervisor means from the said composition means ;
- generating and sending said degradation directive signal to said means while also taking into account said compositor status signal.

56. Composition method according to claim 48 wherein steps (a) to (c) are carried out prior to step (e).

57. Composition method according to any of claims 51 to 55, further comprising the step of sending said complexity information signal to said compositor supervisor means.

58. Decoding method according to any of claims 53 to 56 wherein the step of generating said degradation directive signal consists in associating at least a degradation algorithm to at least one composition algorithm, said composition algorithm and said degradation algorithm being associated to a complexity coefficient/weight representative of the execution time of said composition algorithm and said degradation algorithm.

59. Composition method according to claim 57 wherein said complexity coefficients/weights are dependent on the compositor used for implementing said composition method.

60. Composition method according to claim 57 or 58, wherein said composition time prediction signal is a function of

Aylo Exhibit 1002
Page 282 of 1405

the occurrence of use of said composition algorithms and said complexity coefficients/weights according to the following equation :

$$T(t) \cong T_{Pred}(t) = \sum_{i=1}^{I} c_i w_i \quad (9),$$

where T is the time necessary to compose a audio/video element, T_{Pred} is the composition time prediction of said audio/video element, c_i is the occurrence of said composition algorithm, and w_i is the complexity coefficient/weight.

61. Composition method according to claim 59, wherein when the composition time prediction is a function of the occurrence of use of said composition algorithms and degradation algorithms, said degradation algorithms being associated to a complexity coefficient/weight.

62. Composition method according to claim 59 or 60, wherein when the composition time prediction is greater than the allocated composition time, one uses a degradation method according to which one selects the degradation algorithm instead of its associated composition algorithm according to equation :

$$T_{process} = \sum_{i \in S_{CGD}} (a_i w_i^{CGD} + b_i w_i) + \sum_{i \in S_{Orig}} c_i w_i \leq T_{Alloc} \quad (13)$$

where, $T_{Process}$ is the prediction time necessary to compose a audio/video element, said composition time prediction taking account the degradation, c_i is the occurrence of composition algorithm, a_i is the number the selected degradation algorithms, w_i^{CGD} is the complexity coefficient/weight of the degradation algorithm, b_i is the number of the non selected degradation algorithms, and w_i is the complexity coefficient/weight, where $a_i + b_i = c_i$, so as to reach a $T_{Process}$ lesser than or equal to said allocated composition time.

63. Composition method according to claim 60 or 61, wherein said composition status signal is also used for monitoring the current composition time of a composition video/audio element and make sure that said current composition time is in conformity with the associated composition time prediction.

64. Composition method according to claim 62, wherein if said current composition time is not in conformity with the associated composition time prediction, then said composition supervisor means uses at least one further degradation algorithm defined in claim 61.

65. Composition method according to claim 63 wherein if said current composition time is not in conformity with the associated composition time prediction, then said decoder supervisor means further uses said complexity information signal.

66. Composition method according to any of claims 51 to 63 for a plurality of audio/video composition elements wherein when the composition time prediction is the sum of the composition prediction time of each of said audio/video composition elements, said composition time prediction being defined in claim 50.

67. Composition system for an uncompressed digital video element including video frames and/or video planes and/or video objects, textures, texts, 2 and 3 dimensional graphic primitives, 2 and 3 dimensional models, comprising :

- a video compositor receiving said video element on one input terminal, said compositor being intended to compose said video element using a composition syntax using a plurality of composing algorithms, and to provide, on one output terminal, a representation in a common video representation space said video element ;
- means for determining statistic of use of at least one of said composition algorithms,
- and means for generating a digital statistic signal representative of said statistics for each of said composed video element.

68. Composition system for an uncompressed digital audio element including audio packets, audio objects, synthetic audio objects, comprising :

Aylo Exhibit 1002
Page 283 of 1405

- a audio compositor receiving said audio element on one input terminal, said compositor being intended to compose said audio element using a composition syntax using a plurality of composing algorithms, and to provide, on one output terminal, a representation in a common audio representation space said audio element ;
- means for determining statistic of use of at least one of said composition algorithms,
- and means for generating a digital statistic signal representative of said statistics for each of said composed audio element.

69. Composition system for an audio-visual object comprising a composition system according to claim 66 combined with a composition system according to claim 67.

Aylo Exhibit 1002
Page 284 of 1405

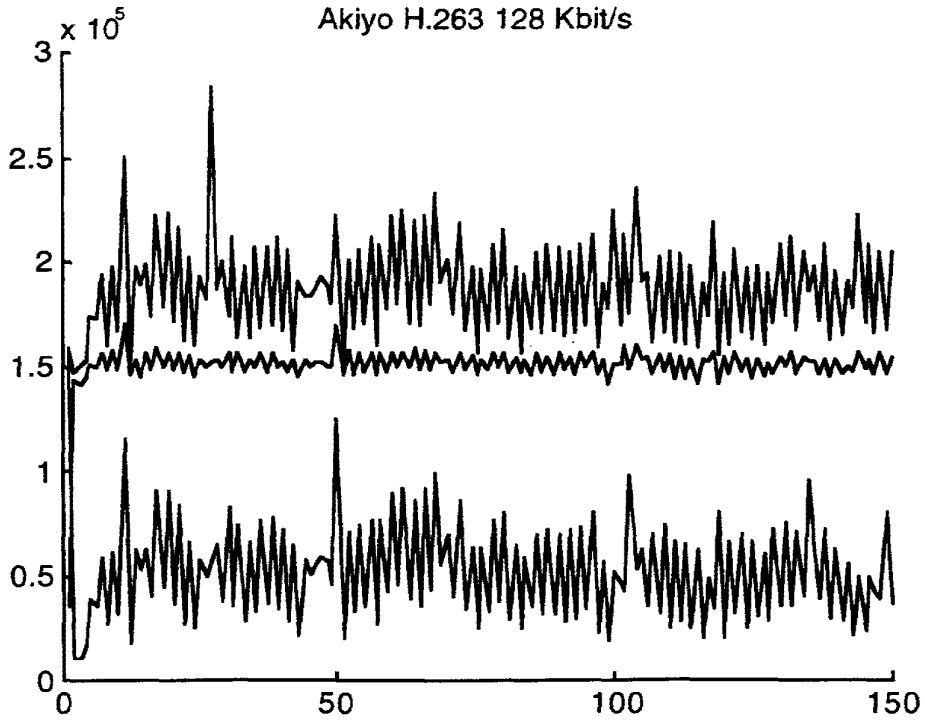


Figure 1

Aylo Exhibit 1002
Page 285 of 1405

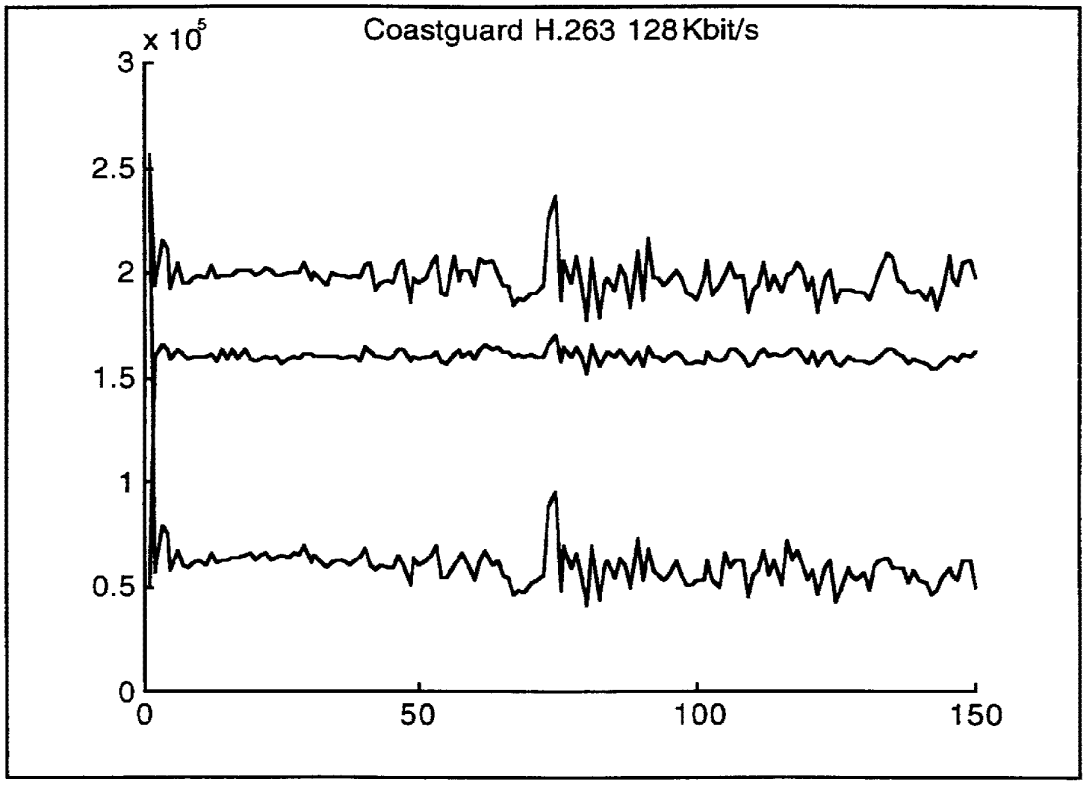


Figure 2

Aylo Exhibit 1002
Page 286 of 1405

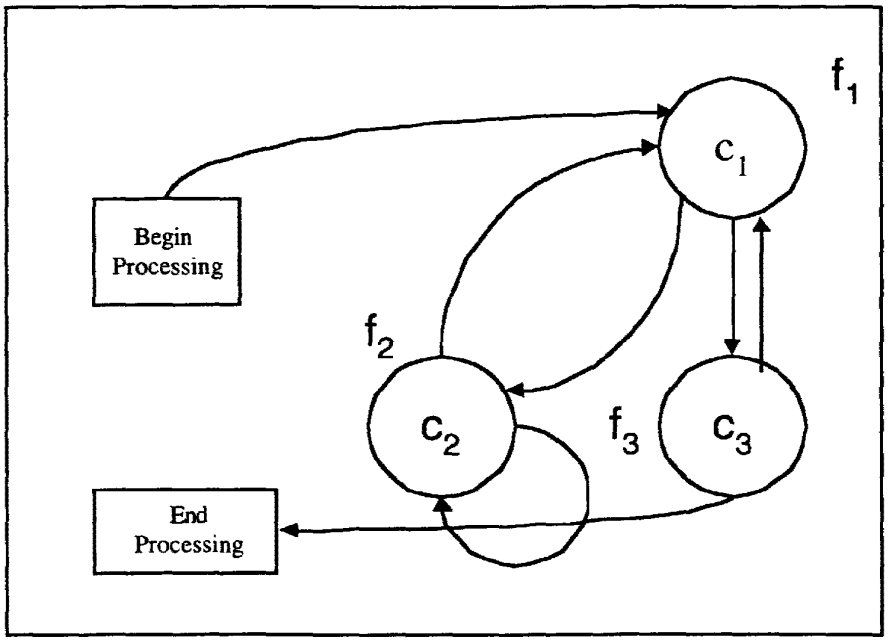


Figure 3

Aylo Exhibit 1002
Page 287 of 1405

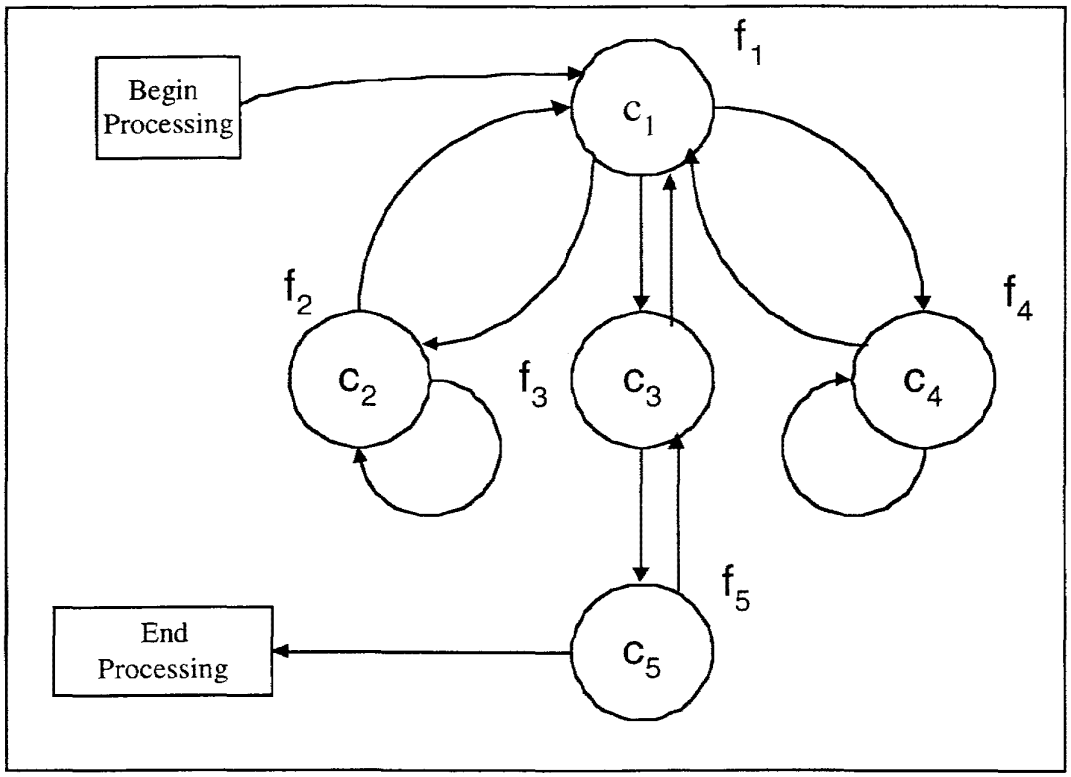


Figure 4

Aylo Exhibit 1002
Page 288 of 1405

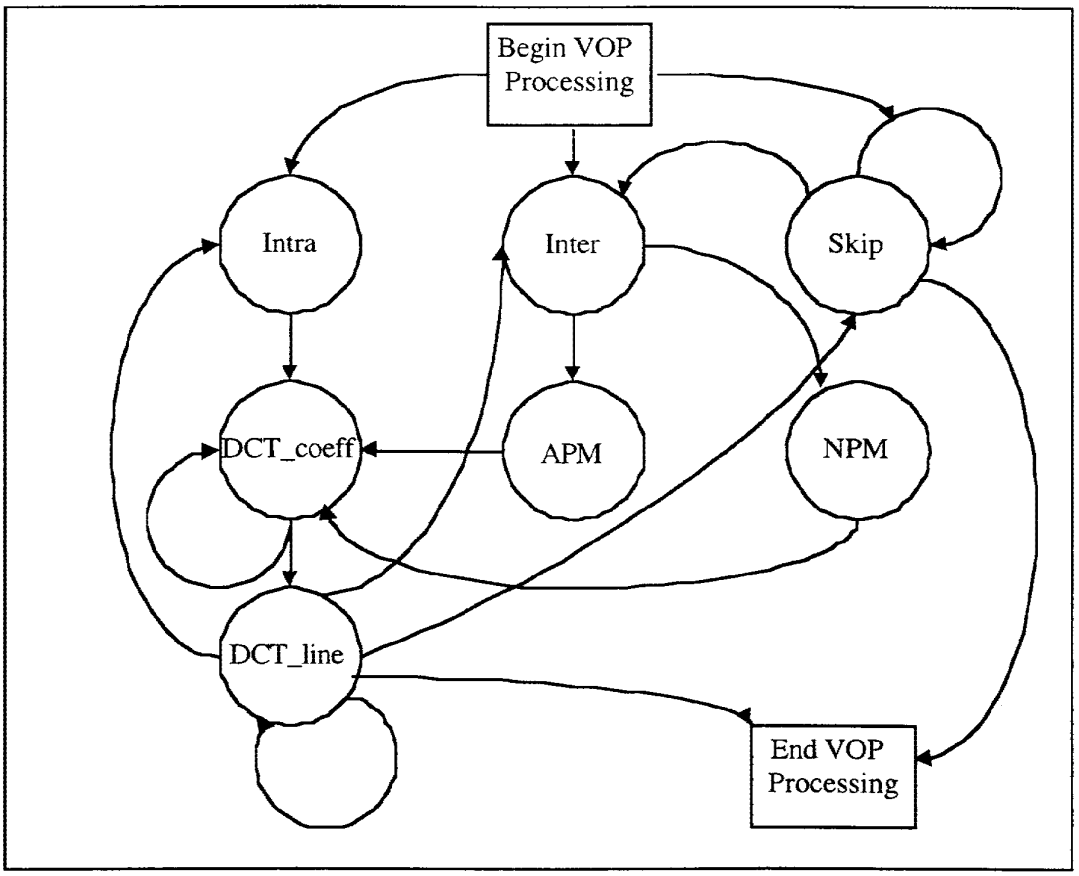


Figure 5

Aylo Exhibit 1002
Page 289 of 1405

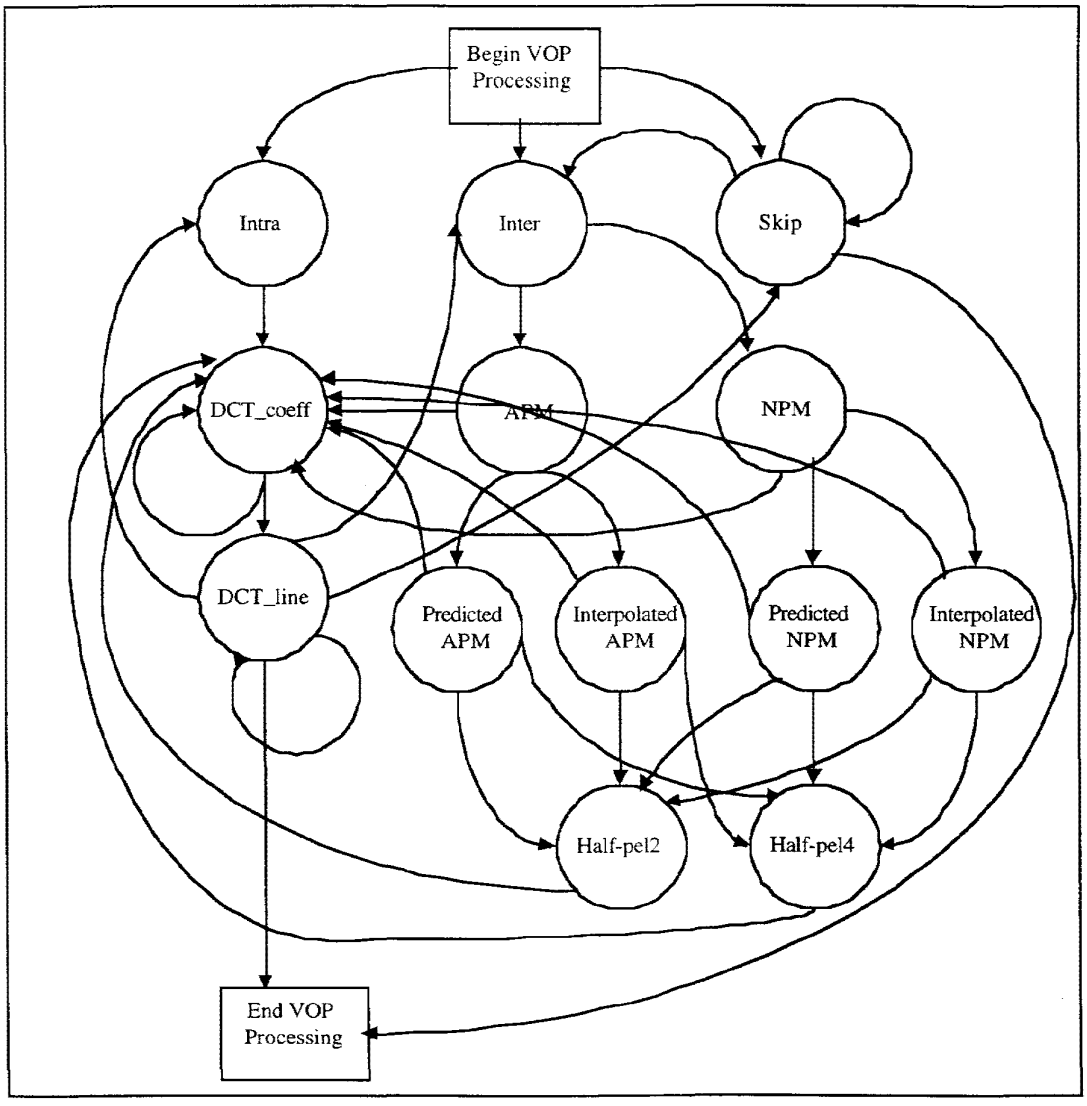


Figure 6

Aylo Exhibit 1002
Page 290 of 1405

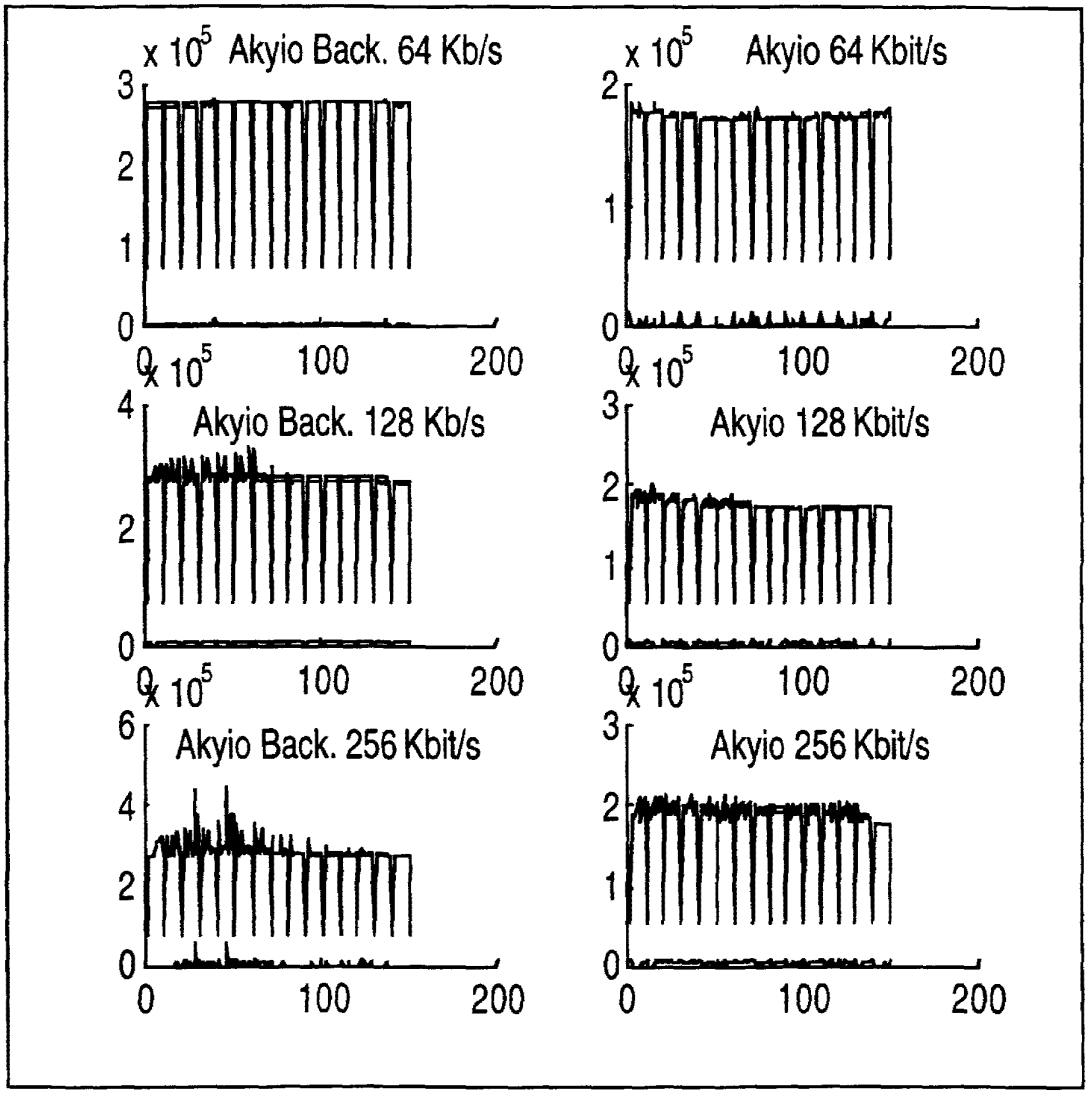


Figure 7

Aylo Exhibit 1002
Page 291 of 1405

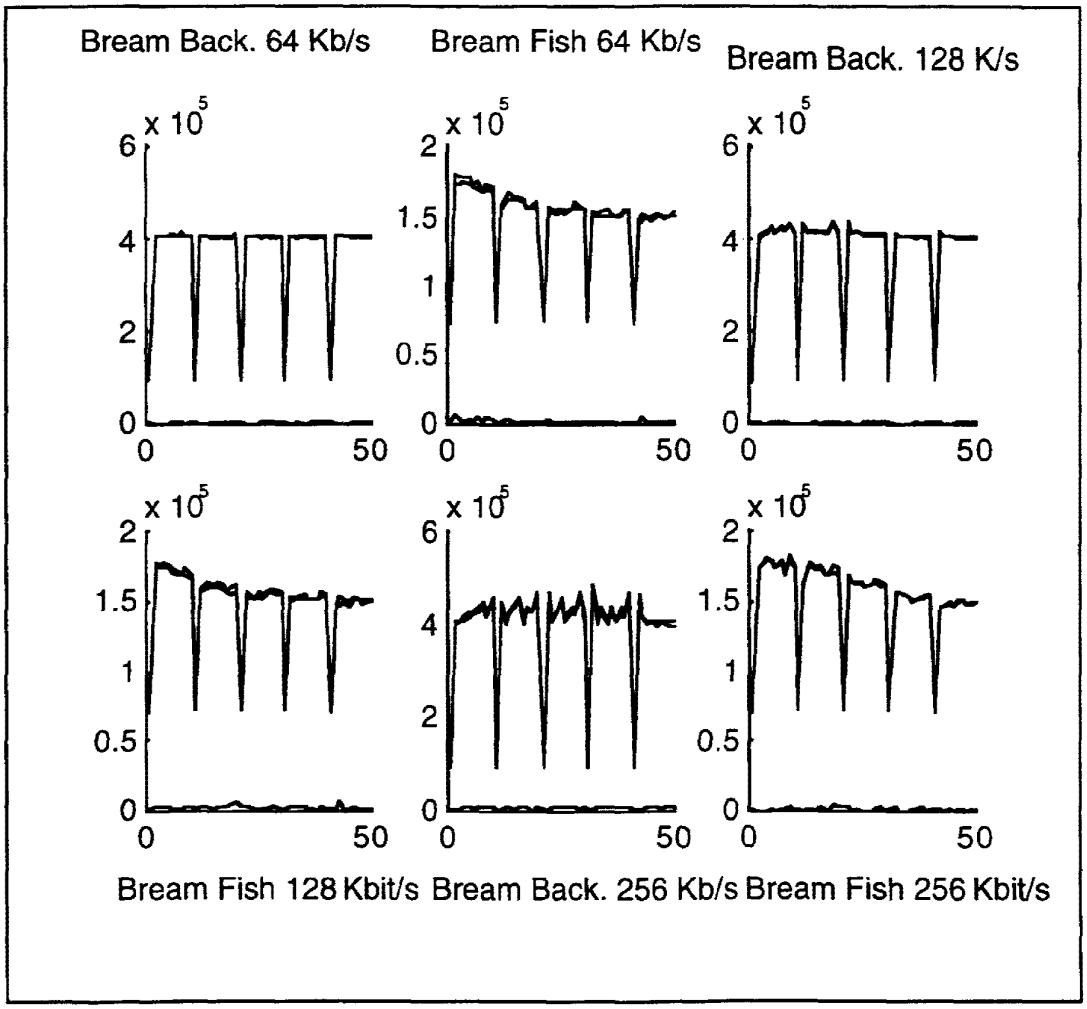


Figure 8

Aylo Exhibit 1002
Page 292 of 1405

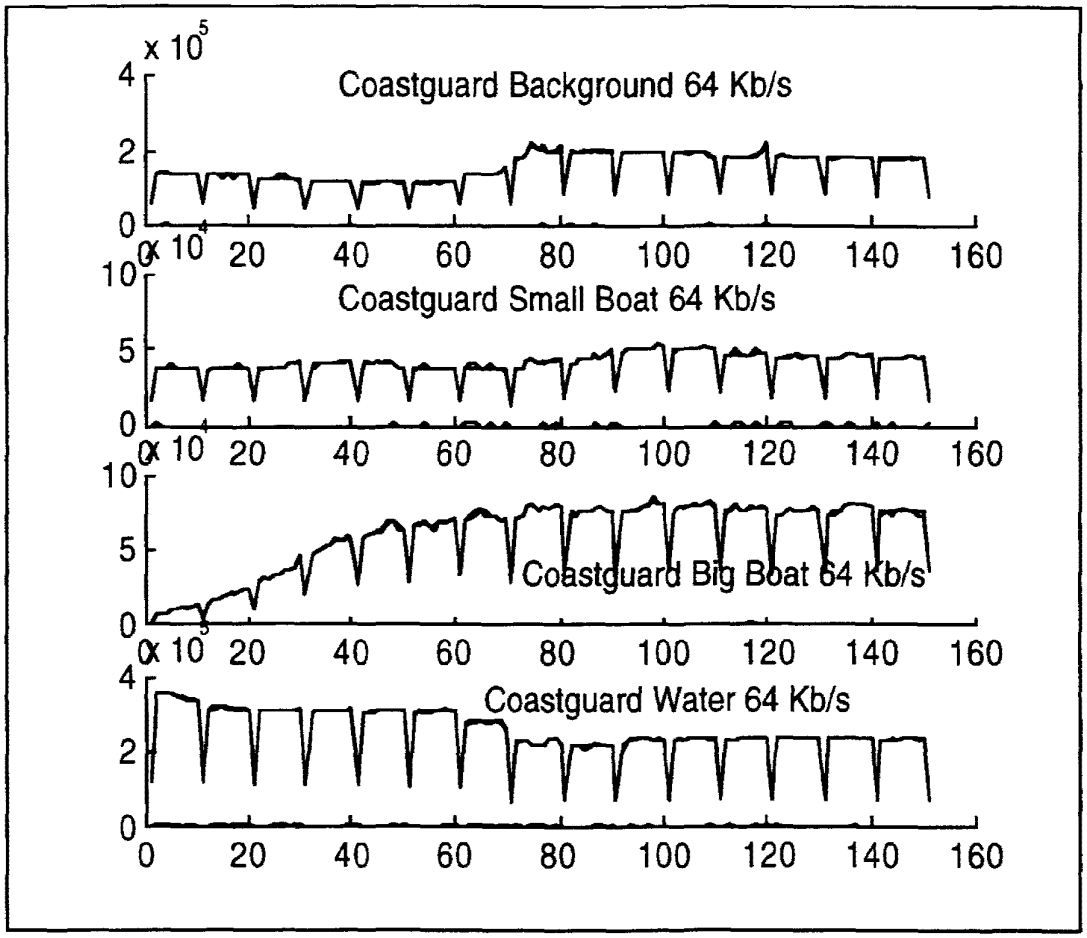


Figure 9

Aylo Exhibit 1002
Page 293 of 1405

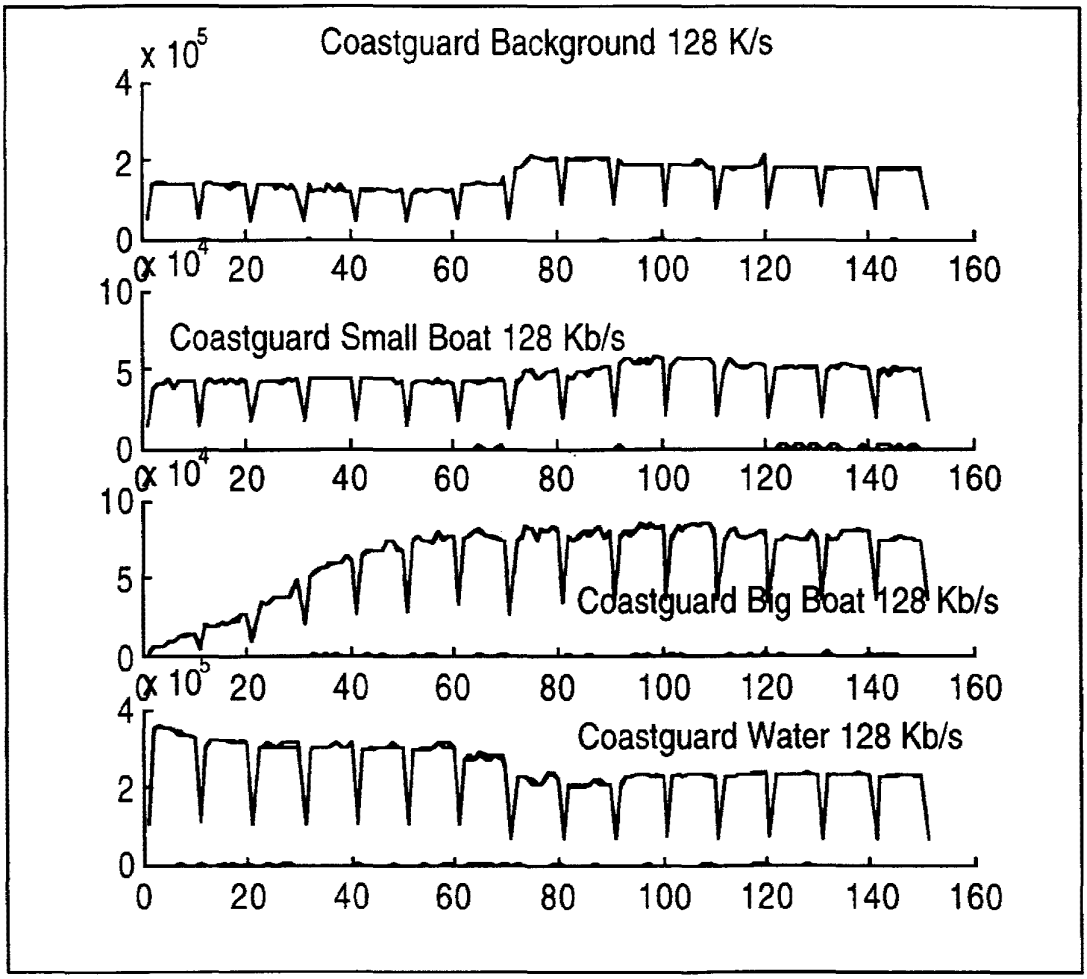


Figure 10

Aylo Exhibit 1002
Page 294 of 1405

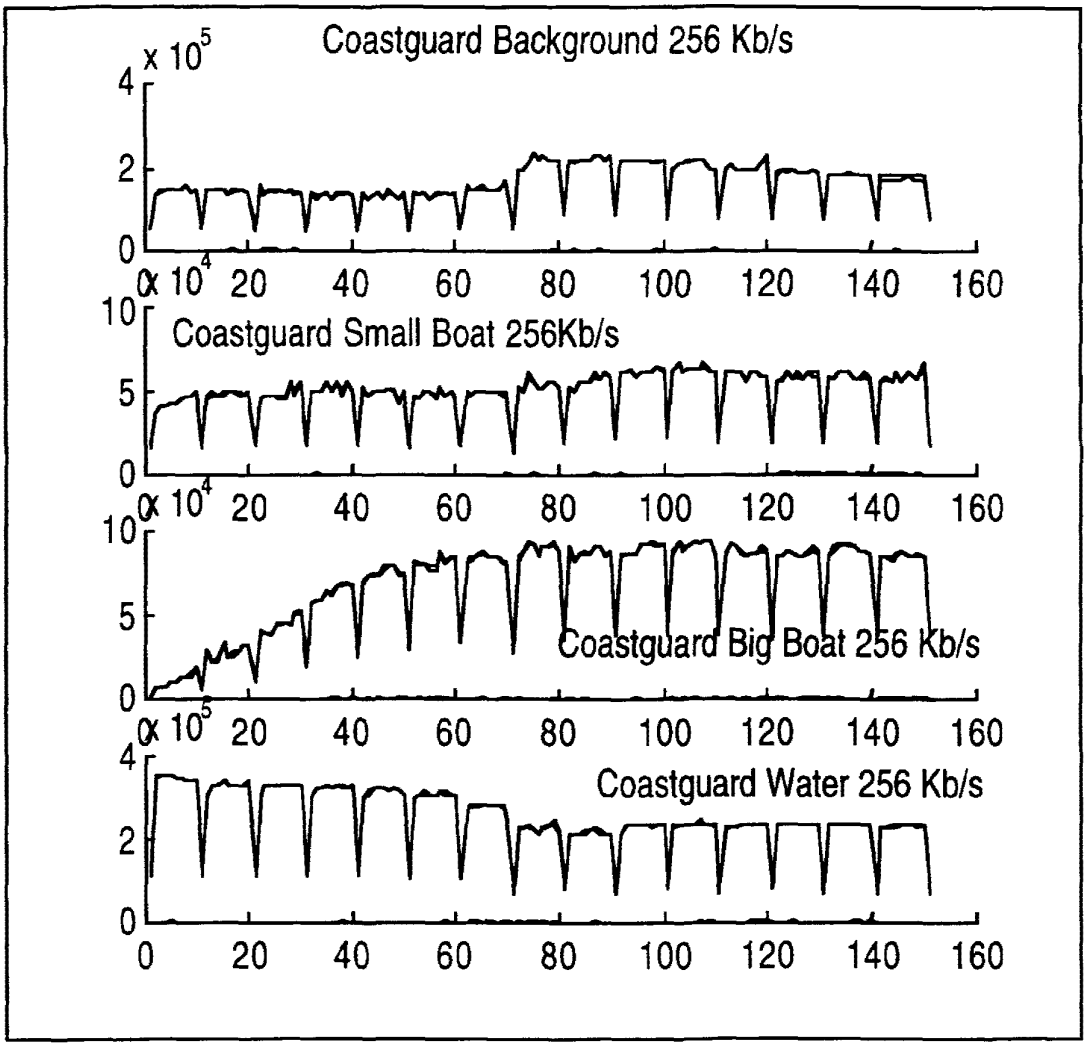


Figure 11

Aylo Exhibit 1002
Page 295 of 1405

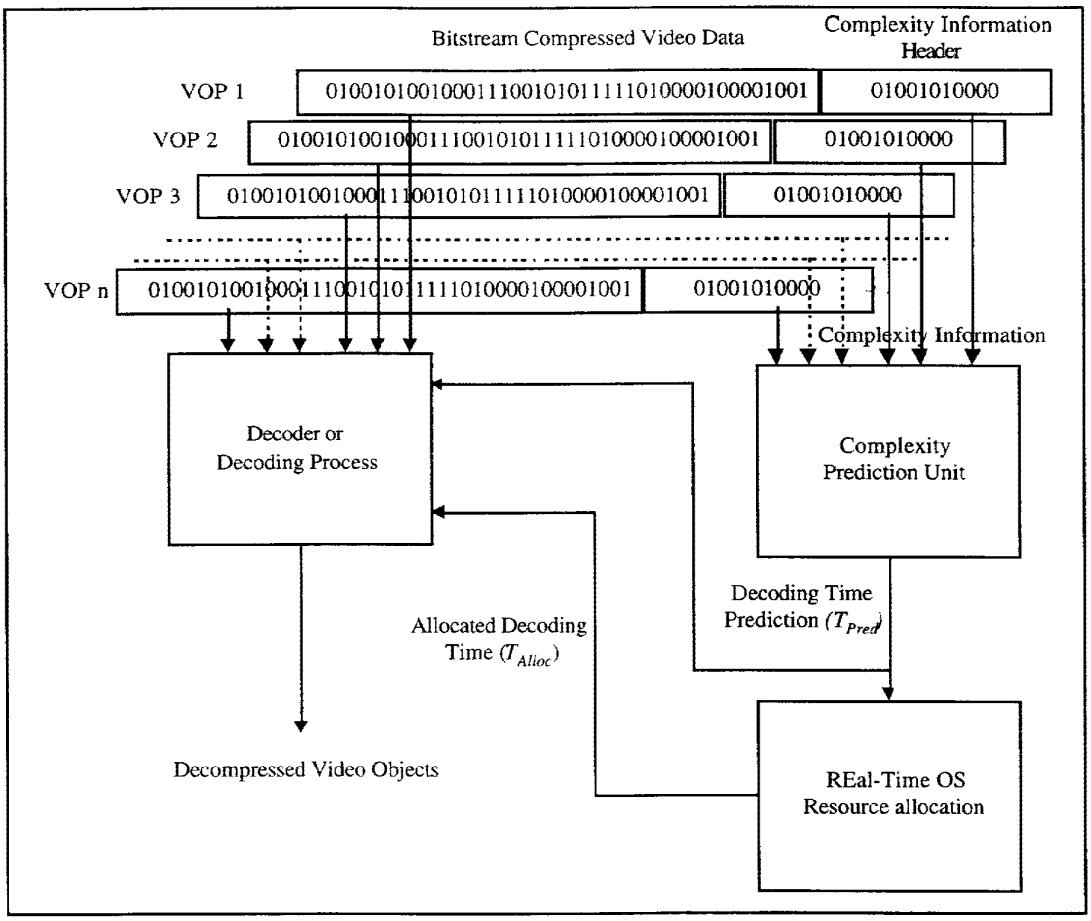


Figure 12

Aylo Exhibit 1002
Page 296 of 1405

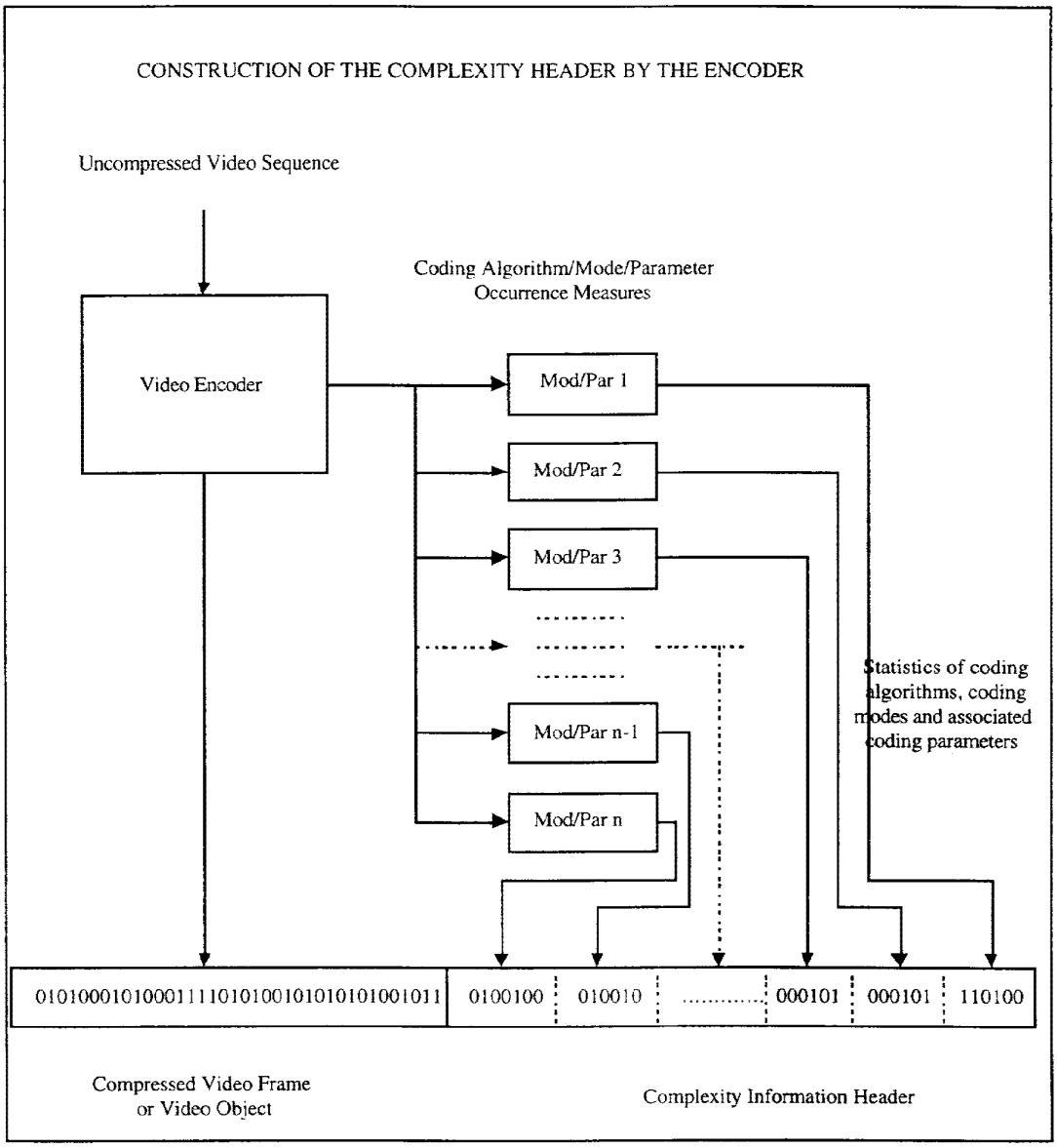


Figure 13

Aylo Exhibit 1002
Page 297 of 1405

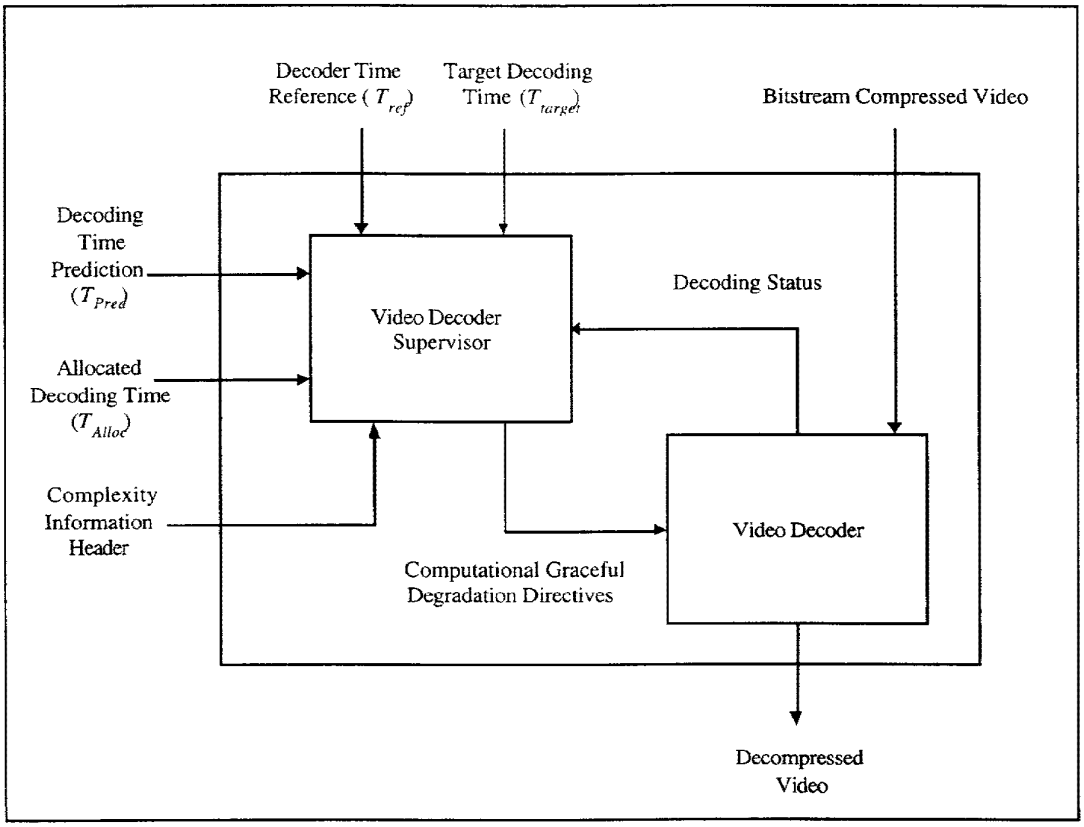


Figure 14

Aylo Exhibit 1002
Page 298 of 1405

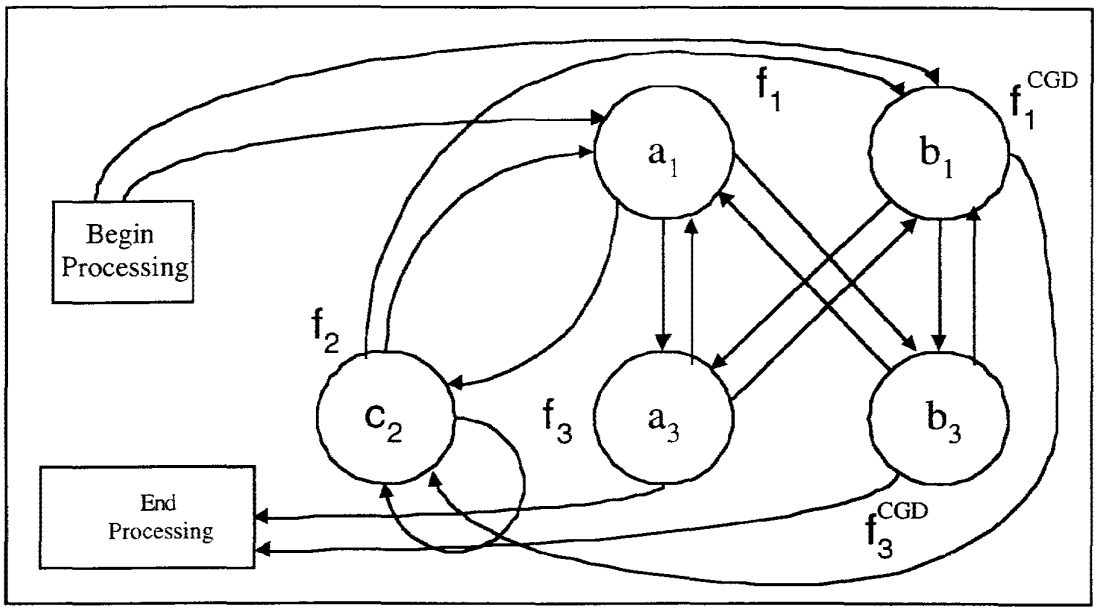

Begin Processing
End Processing
a_1
b_1
c_2
a_3
b_3
f_1
f_1^{CGD}
f_2
f_3
f_3^{CGD}

Figure 15

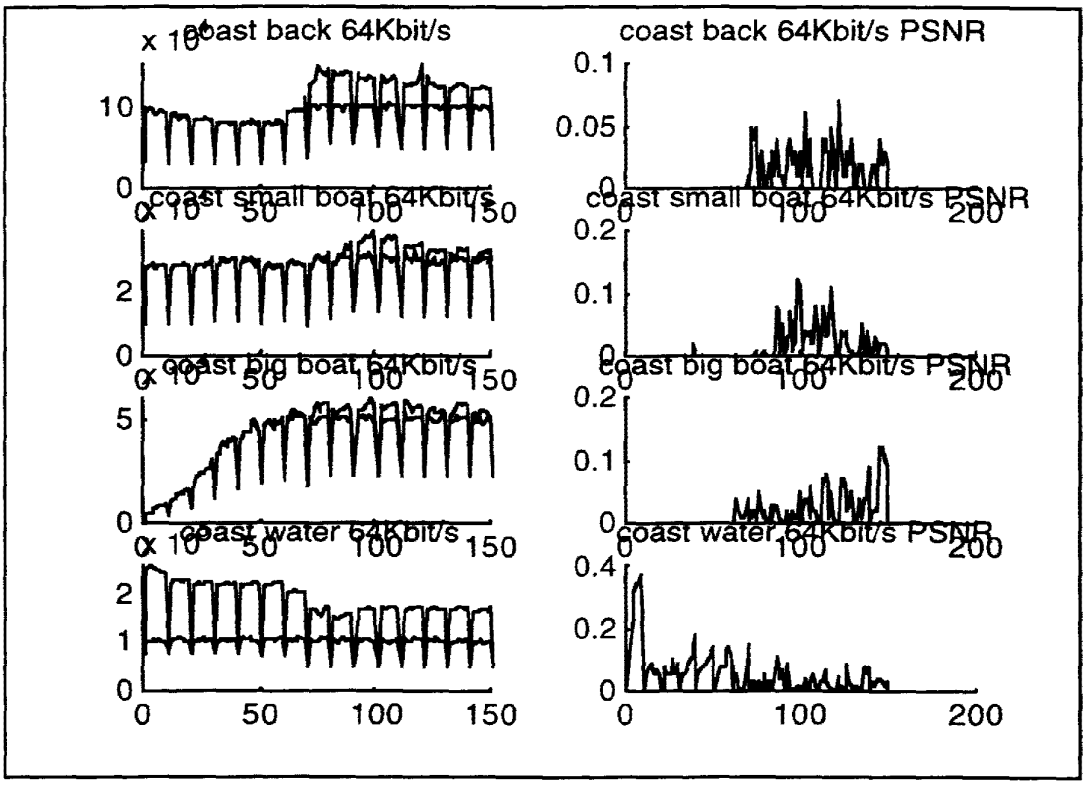


Figure 16

Aylo Exhibit 1002
Page 300 of 1405

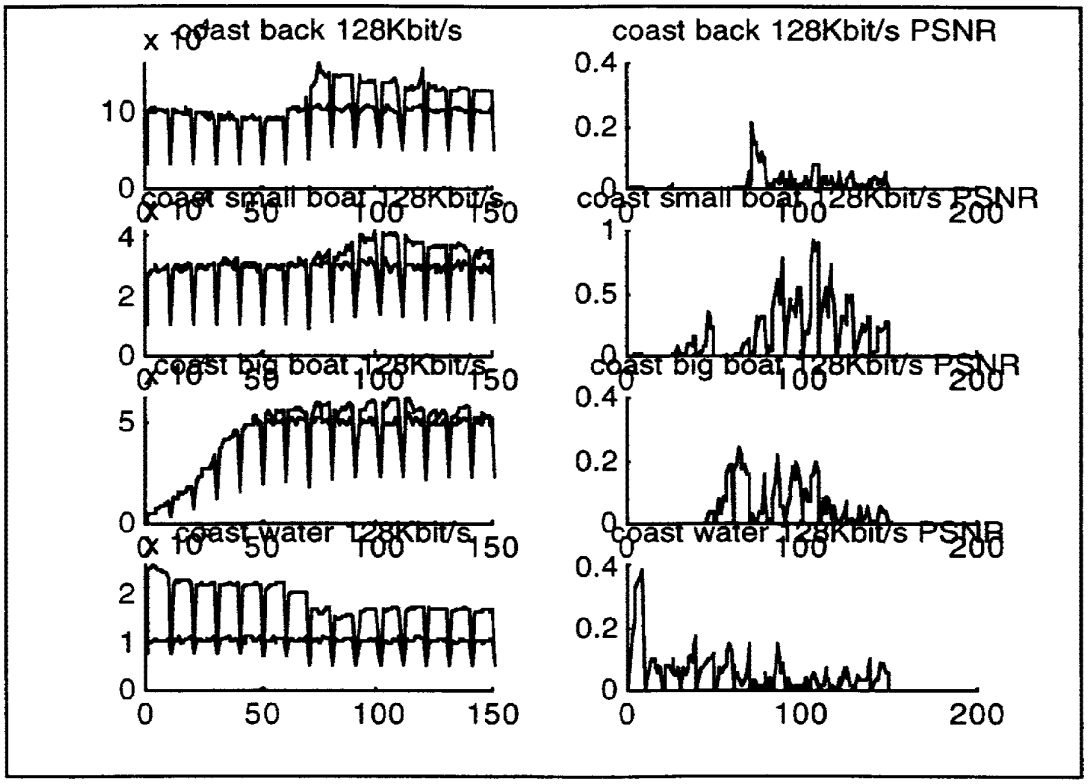


Figure 17

Aylo Exhibit 1002
Page 301 of 1405

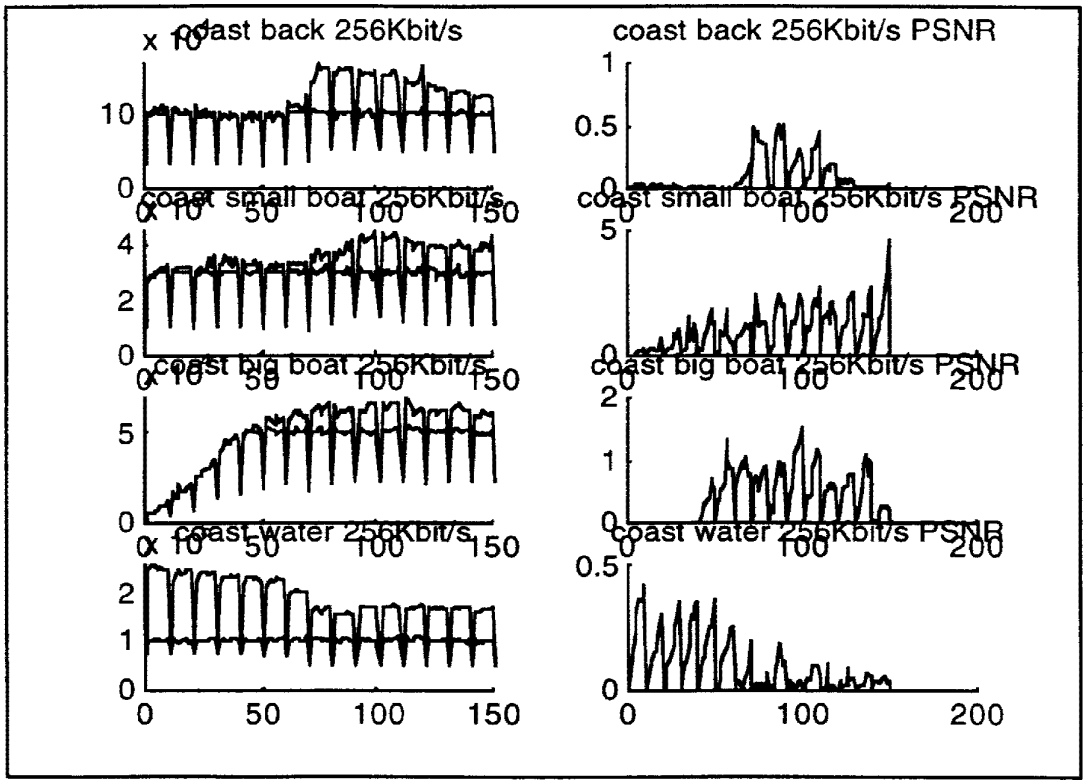


Figure 18

Aylo Exhibit 1002
Page 302 of 1405

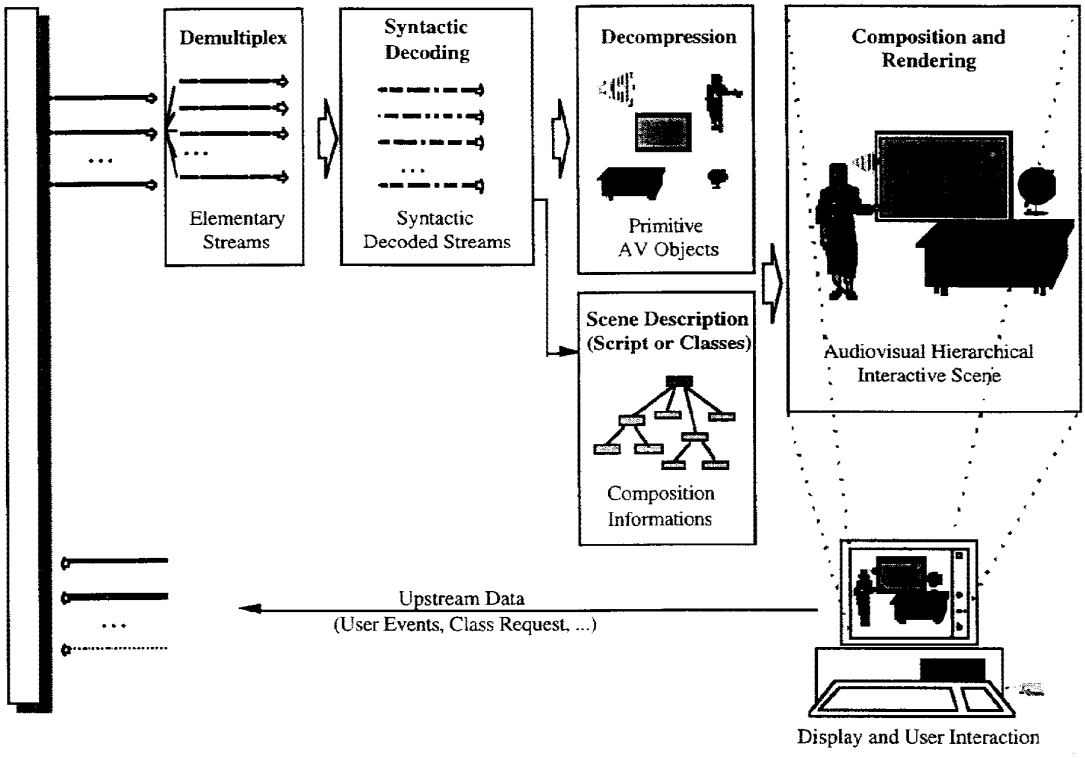


Figure 19

Aylo Exhibit 1002
Page 303 of 1405

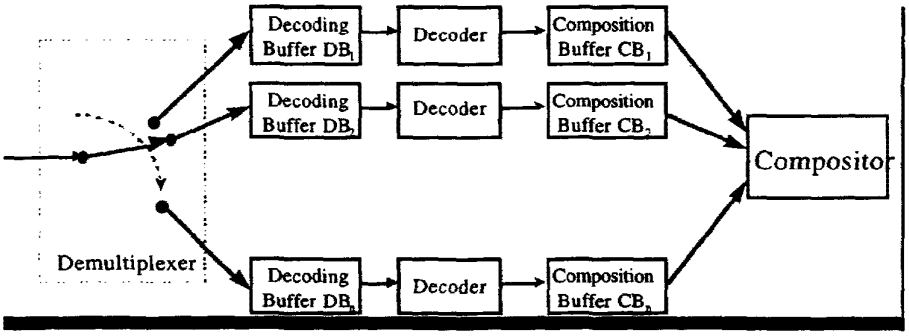

Decoding Buffer DB1
Decoder
Composition Buffer CB1
Decoding Buffer DB2
Decoder
Composition Buffer CB2
Compositor
Demultiplexer
Decoding Buffer DBn
Decoder
Composition Buffer CBn

Figure 20

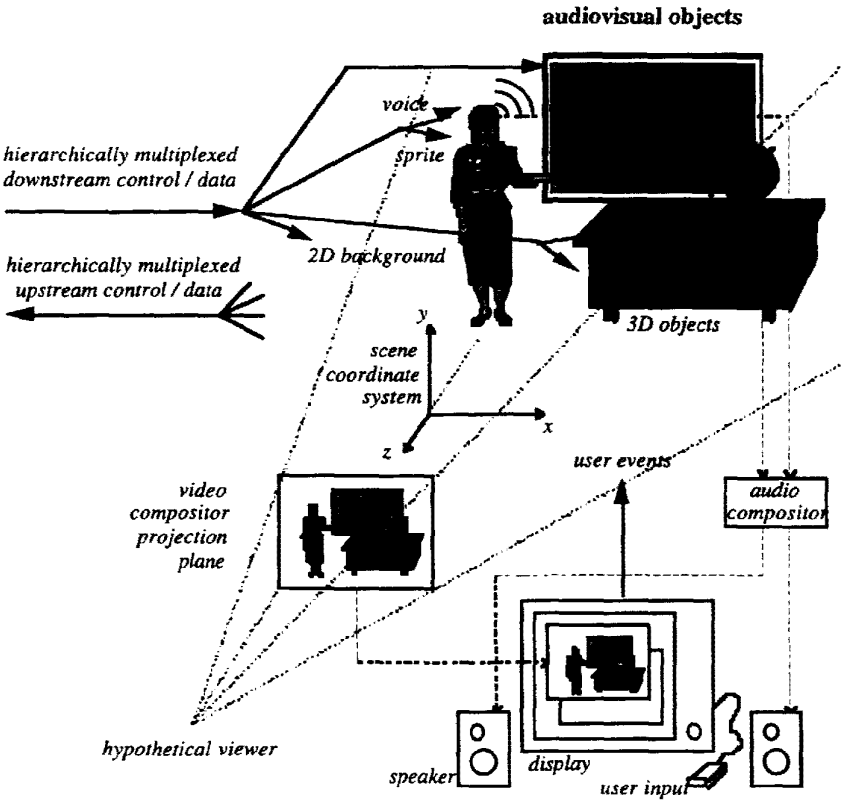


Figure 21

Aylo Exhibit 1002
Page 305 of 1405

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 98 10 9423

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)
E	EP 0 866 426 A (ALSTHOM CGE ALCATEL) 23 September 1998 * abstract; claims 1,5,6 *	1-68	G06T9/00
A	EP 0 512 174 A (SEMAPHORE INC) 11 November 1992 * page 2, line 40 - line 43 * * page 3, line 32 - line 36 *	1-68	
A	MATTAVELLI M ET AL: "Computational graceful degradation for video sequence decoding" PROCEEDINGS. INTERNATIONAL CONFERENCE ON IMAGE PROCESSING (CAT. NO.97CB36144), PROCEEDINGS OF INTERNATIONAL CONFERENCE ON IMAGE PROCESSING, SANTA BARBARA, CA, USA, 26-29 OCT. 1997, pages 330-333 vol.1, XP002080307 ISBN 0-8186-8183-7, 1997, Los Alamitos, CA, USA, IEEE Comput. Soc, USA		

TECHNICAL FIELDS SEARCHED (Int.Cl.6)

G06T

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
THE HAGUE	12 October 1998	Pierfederici, A

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)

Aylo Exhibit 1002
Page 306 of 1405

Electronic Acknowledgement Receipt

EFS ID:	41130700
Application Number:	16876579
International Application Number:	
Confirmation Number:	3720
Title of Invention:	APPARATUS, SYSTEM, AND METHOD FOR MULTI-BITRATE CONTENT STREAMING
First Named Inventor/Applicant Name:	DAVID F. BRUECK
Customer Number:	70560
Filer:	Brett Carlson/Julie Graf
Filer Authorized By:	Brett Carlson
Attorney Docket Number:	066.0141X1C9
Receipt Date:	17-NOV-2020
Filing Date:	18-MAY-2020
Time Stamp:	15:15:54
Application Type:	Utility under 35 USC 111(a)

Payment information:

Submitted with Payment	yes
Payment Type	CARD
Payment was successfully received in RAM	$260
RAM confirmation Number	E2020AGF17106225
Deposit Account	
Authorized User	
The Director of the USPTO is hereby authorized to charge indicated fees and credit any overpayment as follows:	

File Listing:

Document Number	Document Description	File Name	File Size(Bytes)/ Message Digest	Multi Part /.zip	Pages (if appl.)
1	Transmittal Letter	20201116_IDSCert_066-0141X1C9.pdf	227440 176ba1014fc671051cf91a994df4b3c9296caea5	no	1
Warnings:					
Information:					
2	Foreign Reference	CA2466482A1.pdf	1503485 c866a2b6c8e5f6966e9cca39bffff0b45b2ea0b2	no	42
Warnings:					
Information:					
3	Foreign Reference	EP0919952A1.pdf	2283304 9233a93ab4caae309974a4e6475c5b543a906d93	no	49
Warnings:					
Information:					
4	Foreign Reference	EP1202487A2.pdf	1169542 e3961c13249effa85f97034caddedff4dbedc77d	no	21
Warnings:					
Information:					
5	Foreign Reference	EP1298931A2.pdf	564750 17988c97c90278abafab90acccb307ac133ea832	no	9
Warnings:					
Information:					
6	Foreign Reference	EP1395014A1.pdf	430935 5bd3cb3cce066c412e84ceb0db005d86ebc13605	no	11
Warnings:					
Information:					

7	Foreign Reference	EP1670256A2.pdf	2755428 c56ab908b009cc2e63418387ee77d04d92855e0b	no	43
Warnings:					
Information:					
8	Foreign Reference	EP1777969A1.pdf	549893 e15832faf97a870cbe3e74fb08ab268c87af9ec7	no	9
Warnings:					
Information:					
9	Foreign Reference	GB2367219A.pdf	1022882 4b911a530138606f2eb7560e4982065ed5cfef6d	no	24
Warnings:					
Information:					
10	Foreign Reference	JP2000201343.pdf	21958041 852daf4ae4a533b5a36984e992609dbc8888bebb	no	14
Warnings:					
Information:					
11	Foreign Reference	JP2001092752.pdf	9449147 3115b20d4cc1a9898d8742728f2946d9e91da1b7	no	12
Warnings:					
Information:					
12	Foreign Reference	JP2011004225A.pdf	2693988 c96187633b1948096c41bf2f2558ceb94073c945	no	11
Warnings:					
Information:					
13	Non Patent Literature	NPL_Fujisawa_Implementation OfEfficientAcdess.pdf	511563 91dff145bd502cbcca14ed40adf4002af21abab7	no	10
Warnings:					
Information:					

14	Non Patent Literature	NPL_Guo_DirectStream.pdf	7958674 612e3ca1a57a5f239df3d1d3d3235b392d47e87b	no	22
Warnings:					
Information:					
15	Non Patent Literature	NPL_Kozamernik_MediaStreaming.pdf	6407225 7f5c49521521198a912e04f054f8405ed84f9614	no	15
Warnings:					
Information:					
16	Non Patent Literature	NPL_Lienhart_DistributedVideo.pdf	11245900 37342a4029d40ede417d94095bbd90befa4931d5	no	29
Warnings:					
Information:					
17	Non Patent Literature	NPL_Liu_AdaptiveVideoMulticast.pdf	5897378 6d1568a810d08f4e566bfcc4b02748abfa471b64	no	12
Warnings:					
Information:					
18	Non Patent Literature	NPL_Rejaie_ArchitecturalConsiderations.pdf	3252815 d638402b4b84964353e12f7de0a6dc071af47d63	no	6
Warnings:					
Information:					
19	Non Patent Literature	NPL_Roy_SystemArchitecture.pdf	2812918 7e9c5ee17749c18ea0bd37428a2cfaa78b62fddc	no	6
Warnings:					
Information:					
20	Non Patent Literature	NPL_Tsuru_RecentEvolution.pdf	1522645 c5225e2e26324cd4748599f0753210c21c47441b	no	9
Warnings:					
Information:					

21	Non Patent Literature	NPL_WeblioDictionaryPgs.pdf	525676 eec0a963413300c3b5fb79ab7eacbae55666efc2	no	3
Warnings:					
Information:					
22	Non Patent Literature	NPL_Xu_PeerToPeer.pdf	4454232 01dcd4f73c5fcba31a2d951f592bf59eaebca748	no	10
Warnings:					
Information:					
23	Non Patent Literature	NPL_Zhang_CoolStreaming.pdf	293506 79257b4aeb9b9510bf46f97c437f91285abb9784	no	14
Warnings:					
Information:					
24	Foreign Reference	WO2001067264.pdf	1561090 14ea9fe21f267c32636564e07838fe524ecef3c8	no	41
Warnings:					
Information:					
25	Foreign Reference	WO2004025405.pdf	825266 e160c7648ab3c92ae53aad7efe41ffe5c6b0ebc9	no	20
Warnings:					
Information:					
26	Foreign Reference	WO2006010113.pdf	5663204 6e292580427daf9fc8f17d85c0318e226cf0bd6a	no	118
Warnings:					
Information:					
27	Non Patent Literature	Huang_AdaptiveLiveVideoStreaming.pdf	2524324 d11b5abc7e3870ffc0f0d6176396674bdbbc8c7a	no	11
Warnings:					
Information:					

28	Non Patent Literature	Krasic_QoSScalabilityForStreamedMediaDelivery.pdf	232474 63f916fa208fe954d7b15a25ec0c3c3ec7c3704b	no	10
Warnings:					
Information:					
29	Non Patent Literature	Krasic_Quality-AdaptiveMedia.pdf	288035 2c5a1fb17fd9783462bcea4b14497ec76424909b	no	10
Warnings:					
Information:					
30	Non Patent Literature	Walpole_APlayerForAdaptiveMPEG.pdf	1119882 ddb4cc0ec5db2c359797f0d721cff194e6c1b91d	no	12
Warnings:					
Information:					
31	Non Patent Literature	NPL_Nguyen_MultipleSenderDistVideo.pdf	1053500 af97412420b4c476c2c59e0e76443e17141b1bb	no	12
Warnings:					
Information:					
32	Non Patent Literature	NPL_Yoshimura_MobileStreamnigMedia.pdf	378918 aac54a2603ed446586676ed16c15022af1acc1eb	no	20
Warnings:					
Information:					
33	Non Patent Literature	NPL_MoveNetworks_NextGeneration.pdf	1128862 446af274461f93a7d60548a003fcc17a6c5e4a0f	no	21
Warnings:					
Information:					
34	Non Patent Literature	NPL_Clement_MoveClosesFunding.pdf	172934 c65a317075d6d88f73613f212b13b4e09646ea79	no	2
Warnings:					
Information:					

35	Non Patent Literature	NPL_Wicker_ErrorControlSys_Part1.pdf	4017078 05a44138df970b54e4654a8e826e77ee3111a734	no	85
Warnings:					
Information:					
36	Non Patent Literature	NPL_Wicker_Part2.pdf	4003752 9af8bc8ca3ee2444a65208263b785ffdf6f0df42	no	81
Warnings:					
Information:					
37	Non Patent Literature	NPL_Wicker_Part3.pdf	5167479 ca519c14ff8af13af6d790858696235a433b2fcc	no	89
Warnings:					
Information:					
38	Non Patent Literature	NPL_Wicker_Part4.pdf	8345160 4e2a64c21ea5974a4a96928ea79904151b31b1a3	no	95
Warnings:					
Information:					
39	Non Patent Literature	NPL_Wicker_Part5.pdf	16458768 9c07a14f8208b38c97d27b7d16e139b240d12c4f	no	91
Warnings:					
Information:					
40	Non Patent Literature	NPL_Wicker_Part6.pdf	12015062 7d72bb382aa4c68c0f1ed8169dadd6689aaf1fe2	no	68
Warnings:					
Information:					
41	Non Patent Literature	NPL_Goyal_CodingCompression.pdf	4353461 78149b1c9bf070e64d21f2a37ba8f1d66dbd45ff	no	20
Warnings:					
Information:					

42	Non Patent Literature	NPL_Puri_SourceCoding.pdf	395947 bdd0ea033a2ab94cb2477b4e4b363bde71280a6e	no	5
Warnings:					
Information:					
43	Non Patent Literature	NPL_Albanese_PriorityEncoding.pdf	1436896 40ea8d1dc311303a58036aa6803271aebeec3986	no	36
Warnings:					
Information:					
44	Non Patent Literature	NPL_Birney_IntelligentStreaming.pdf	631194 0d23a6e65ccfb6f0463367759c3f890f0a55c6cc	no	7
Warnings:					
Information:					
45	Non Patent Literature	NPL_Pathan_ContentDelivery.pdf	3819974 bd517be17fd62cd8c811c76a32b0a45da2230af0	no	44
Warnings:					
Information:					
46	Non Patent Literature	NPL_On2Technologies.pdf	887186 cc8df69d3dd6cc62a111dc6d213c583e0683bc58	no	13
Warnings:					
Information:					
47	Non Patent Literature	NPL_Liu_PeerToPeerInternetVideo.pdf	1125709 1dbfd7983420ecb8f68943b0539a4d344045f5fb	no	12
Warnings:					
Information:					
48	Non Patent Literature	NPL_Bommaiah.pdf	427349 0120adb7c21f844fbbf72df8893e203595ba229e	no	11
Warnings:					
Information:					

49	Non Patent Literature	NPL_Roy.pdf	203211 6b2c4564d29af008372104a817676f18da1b87c8	no	4
Warnings:					
Information:					
50	Information Disclosure Statement (IDS) Form (SB08)	20201116_IDS-1_066-0141X1C9.pdf	618180 613cb9d71d386079e7bd84fb9c599ea2c292fd7c	no	22
Warnings:					
Information:					
51	Information Disclosure Statement (IDS) Form (SB08)	20201116_IDS-2_066-0141X1C9.pdf	531173 dada9c36f30ab909a5552873adc3a9ec921015be	no	8
Warnings:					
Information:					
52	Information Disclosure Statement (IDS) Form (SB08)	20201116_IDS-3_Form_066-0141X1C9.pdf	615607 824893dd780c5bf0478d06d1a165fad94bf4cea3	no	5
Warnings:					
Information:					
53	Foreign Reference	EP0365683A1.pdf	2561217 37bbada1d374c6bcfea7e968686307e8edb8d56b	no	26
Warnings:					
Information:					
54	Non Patent Literature	NPL_Jadoo-InvalidityAppendix.pdf	419549 534a2db0ec404eb0cea59e11846261a5e336051a	no	54
Warnings:					
Information:					
55	Non Patent Literature	NPL_Jadoo-InvalidityPleading.pdf	515765 b562c223b19dc6dd21f200321f97f3e918b745d4	no	6
Warnings:					
Information:					

56	Non Patent Literature	NPL_Balk_Adaptive-video-streaming--.pdf	337388 978c19345506d0bdf43e75cbf287d77c29be9b11	no	6
Warnings:					
Information:					
57	Fee Worksheet (SB06)	fee-info.pdf	30605 2f734dcda84af834106cd2fe42f6b3336175c129	no	2
Warnings:					
Information:					
			Total Files Size (in bytes):	173357566	

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.

IDS CERTIFICATION STATEMENTS

I. **THIS IDS IS BEING FILED UNDER 37 C.F.R. 1.97(b): (check one box)**

☐ Within three months of the filing date of a national application other than a continued prosecution application under 1.53(d).

☐ Within three months of the date of entry of the national stage as set forth in 1.491 in an international application.

☐ Before the mailing date of a first Office Action on the merits. In the event that a first Office Action on the merits has been mailed, please consider this IDS under 37 C.F.R. 1.97(c) and see the statement under 1.97(e) or charge deposit account 50-2091 the fee specified by 37 C.F.R. 1.17(p) if such statement is not selected in Section IV.

☐ Before the mailing of a first Office Action after the filing of a request for continued examination under 1.114.

II. **THIS IDS IS BEING FILED UNDER 37 C.F.R. 1.97(c):**

☑ After the period specified in 37 C.F.R. 1.97(b) and before the mailing date of any of a final action under 1.113, a notice of allowance under 1.311 or an action that otherwise closes prosecution in the application, and accompanied by:

- ☐ a statement specified in 1.97(e) as subsequently provided in this IDS; or
- ☑ the fee set forth in 1.17(p).

III. **THIS IDS IS BEING FILED UNDER 37 C.F.R. 1.97(d):**

☐ After the period specified in 37 C.F.R. 1.97(c) and on or after the mailing date of any of a final action under 1.113, a notice of allowance under 1.311 or an action that otherwise closes prosecution in the application, but before or simultaneous with payment of the issue fee, and accompanied by a statement specified in 1.97(e) as subsequently provided in this IDS **and** the fee set forth in 1.17(p).

IV. **Statement under 37 C.F.R. 1.97(e) (check only one box)**

The undersigned hereby states that:

☐ Each item of information contained in the IDS was first cited in any communication from a foreign patent office in a counterpart foreign application not more than three months prior to the filing of this IDS.

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

V. **Statement under 37 C.F.R. §1.704(d)**:

☐ The undersigned hereby states that each item of information contained in the information disclosure statement: (i) was first cited in any communication from a patent office in a counterpart foreign or international application or from the Office, and this communication was not received by any individual designated in § 1.56(c) more than thirty days prior to the filing of the information disclosure statement; or (ii) is a communication that was issued by a patent office in a counterpart foreign or international application or by the Office, and this communication was not received by any individual designated in § 1.56(c) more than thirty days prior to the filing of the information disclosure statement..

VI. **CONCLUSION**

It is respectfully submitted that the claims presently on file patently distinguish the present invention from each of the references cited in this IDS. The references cited in this IDS are being cited only in the interest of candor and without any admission that they constitute statutory prior art or contain matter which anticipates the invention or which would render the same obvious, either singly or in a combination, to a person of ordinary skill in the art. Furthermore the filing of this IDS should not be construed to be an admission that the information cited in this IDS is, or is considered to be, material to patentability as defined in 1.56(b). In addition, the filing of this IDS should not be construed as a representation that a search has been made.

If the Examiner has any questions concerning this IDS, the Examiner is requested to contact the undersigned at the telephone number associated with Customer No. 70560 . If it is determined that this IDS has been filed under the wrong rule, the PTO is requested to consider this IDS under the proper rule, with a petition if necessary, and charge the appropriate fee to Deposit Account No. 50-2091.

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2000-201343
(43)Date of publication of application : 18.07.2000

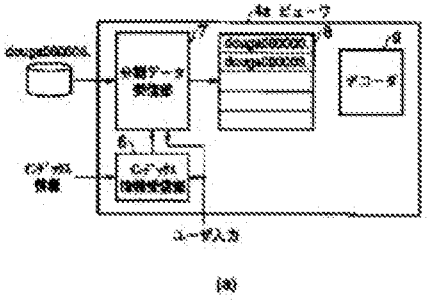
(51)Int.Cl. H04N 7/173

(21)Application number : 11-000497
(22)Date of filing : 05.01.1999

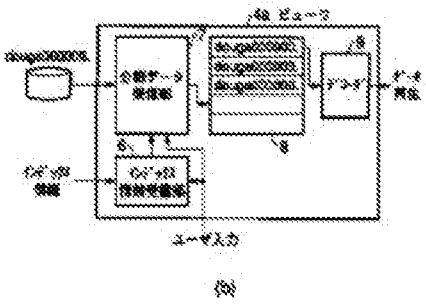
(71)Applicant : TOSHIBA CORP
(72)Inventor : CHIJITANI MASAHIDE

(54) MOVING IMAGE DATA REPRODUCING DEVICE, MOVING IMAGE DATA DIVIDING DEVICE AND RECORDING MEDIUM READABLE BY COMPUTER IN WHICH PROGRAM IS RECORDED

(57)Abstract:
PROBLEM TO BE SOLVED: To enable random access to moving image data without changing a server function.
SOLUTION: A moving image data reproducing device with an index information receiving means 6 to receive index information to be generated by relating plural pieces of divided data to be obtained by deviding the moving image data to its reproduction time information by requesting the information to a server, a divided data receiving means 7 to receive the divided data related to reproduction time information at the reproduction time information to which starting of reproducing is instructed and afterward by successively requesting the information to the server by referring to the index information received by the index information receiving means 6, a storage means 9 to store the divided data received by the divided data receiving means 7 and a reproducing means to successively reproduce the divided data stored in the storage means 9 according to the index information received by the index information receiving means 6 is provided.



(a)



(b)

Aylo Exhibit 1002
Page 318 of 1405

(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2000－201343
（P2000－201343A）
(43)公開日　平成12年７月18日(2000.7.18)

(51)Int.Cl.⁷	識別記号	ＦＩ		テーマコード（参考）
Ｈ０４Ｎ　7/173	６３０	Ｈ０４Ｎ　7/173	６３０	５Ｃ０６４

審査請求　未請求　請求項の数13　ＯＬ　（全 13 頁）

(21)出願番号　特願平11－497

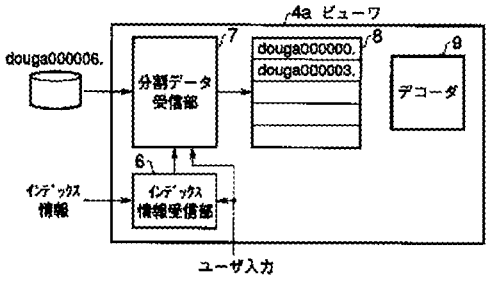
(22)出願日　平成11年１月５日(1999.1.5)

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　千々谷　眞英
東京都府中市東芝町１番地　株式会社東芝府中工場内
(74)代理人　100058479
弁理士　鈴江　武彦　（外６名）

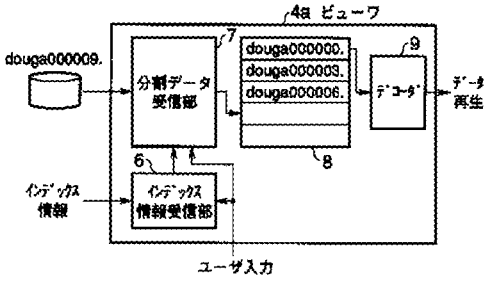
Ｆターム(参考)　5C064 BA07 BB10 BC16 BC20 BC27 BD16

(54)【発明の名称】　動画像データ再生装置及び動画像データ分割装置並びにプログラムを記録したコンピュータ読み取り可能な記録媒体

(57)【要約】
【課題】サーバの機能を変更することなく、動画像データのランダムアクセスを可能にする。
【解決手段】動画像データが分割されて得られる複数の分割データとその再生時刻情報とを関連付けて生成されるインデックス情報を、サーバに要求して受信するインデックス情報受信手段と、当該インデックス情報受信手段に受信されたインデックス情報を参照し、再生開始を指示された再生時刻情報以降の再生時刻情報に関連付けられた分割データをサーバに順次要求して受信する分割データ受信手段と、当該分割データ受信手段に受信された分割データを記憶する記憶手段と、インデックス情報受信手段に受信されたインデックス情報にしたがって、記憶手段に記憶されている分割データを順次再生させる再生手段とを具備した動画像データ再生装置を提供する。



(a)



(b)

1

【特許請求の範囲】

【請求項１】　動画像データが分割されて得られる複数の分割データとそれぞれの再生時刻情報とを関連付けて生成されるインデックス情報を、サーバに要求して受信するインデックス情報受信手段と、

当該インデックス情報受信手段に受信されたインデックス情報にしたがって、再生開始を指示された再生時刻情報及びその後の再生時刻情報に関連付けられた分割データを、サーバに順次要求して受信する分割データ受信手段と、

当該分割データ受信手段によって受信された分割データを記憶する記憶手段と、

前記記憶手段に記憶されている分割データを順次再生させる再生手段とを具備したことを特徴とする動画像データ再生装置。

【請求項２】　請求項１記載の動画像データ再生装置において、

前記分割データの送信をサーバに要求してから、この分割データを受信するまでの時間を推定する時間推定手段を付加し、

前記分割データ受信手段は、当該時間推定手段によって推定された時間とほぼ同じ時間分の分割データが前記記憶手段に記憶されていない場合に、前記インデックス情報にしたがってサーバに分割データを要求し、

前記再生手段は、前記時間推定手段によって推定された時間とほぼ同じ時間分の分割データが前記記憶手段に記憶された場合に、この記憶手段に記憶されている分割データの再生を開始することを特徴とする動画像データ再生装置。

【請求項３】　動画像データが分割されて得られる複数の分割データについて伝送速度が異なる複数の種別が生成されている場合に、この複数種別の各分割データとそれぞれの再生時刻情報とを関連付けて生成されるインデックス情報を、サーバに要求して受信するインデックス情報受信手段と、

前記分割データの送信をサーバに要求してから、この分割データを受信するまでの時間を推定する時間推定手段と、

当該時間推定手段によって推定された時間に基づいて、要求する分割データの種別を選択する分割データ選択手段と、

前記インデックス情報受信手段に受信されたインデックス情報にしたがって、再生開始を指示された再生時刻情報及びその後の再生時刻情報に関連付けられ、かつ前記分割データ選択手段によって選択されている種別の分割データを、サーバに順次要求して受信する分割データ受信手段と、

当該分割データ受信手段によって受信された分割データを記憶する記憶手段と、

前記記憶手段に記憶されている分割データを順次再生させる再生手段とを具備したことを特徴とする動画像データ再生装置。

2

【請求項４】　請求項３記載の動画像データ再生装置において、

前記分割データ受信手段は、前記時間推定手段に推定された時間とほぼ同じ時間分の分割データが前記記憶手段に記憶されていない場合に、前記データ選択手段によって選択されている種別の分割データを、前記インデックス情報にしたがって前記サーバに要求し、

前記再生手段は、前記時間推定手段によって推定された時間とほぼ同じ時間分の分割データが前記記憶手段に記憶された場合に、この記憶手段に記憶されている分割データの再生を開始することを特徴とする動画像データ再生装置。

【請求項５】　動画像データを複数に分割して複数の分割データを生成し、かつ前記複数の分割データとこの分割データの再生時刻情報とを関連付けたインデックス情報を生成する動画像データ分割装置。

【請求項６】　動画像データが分割されて得られる複数の分割データについて伝送速度が異なる複数の種別を生成し、この複数の種別の各分割データをそれぞれの再生時刻情報に関連付けたインデックス情報を生成する動画像データ分割装置。

【請求項７】　コンピュータに、

動画像データが分割されて得られる複数の分割データとそれぞれの再生時刻情報とを関連付けて生成されるインデックス情報を、サーバに要求して受信するインデックス情報受信機能と、

当該インデックス情報受信機能によって受信されたインデックス情報にしたがって、再生開始を指示された再生時刻情報及びその後の再生時刻情報に関連付けられた分割データを、サーバに順次要求して受信する分割データ受信機能と、

当該分割データ受信機能によって受信された分割データを記憶する記憶機能と、

前記記憶機能によって記憶されている分割データを順次再生させる再生機能とを実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項８】　請求項７記載のコンピュータ読み取り可能な記録媒体において、

前記分割データの送信をサーバに要求してから、この分割データを受信するまでの時間を推定する時間推定機能を付加し、

前記分割データ受信機能は、当該時間推定機能によって推定された時間とほぼ同じ時間分の分割データが前記記憶機能によって記憶されていない場合に、前記インデックス情報にしたがってサーバに分割データを要求し、

前記再生機能は、前記時間推定機能によって推定された時間とほぼ同じ時間分の分割データが前記記憶機能によって記憶された場合に、この記憶機能によって記憶され

Aylo Exhibit 1002
Page 320 of 1405

ている分割データの再生を開始することを特徴とするコンピュータ読み取り可能な記録媒体。

【請求項９】　コンピュータに、
動画像データが分割されて得られる複数の分割データについて伝送速度が異なる複数の種別が生成されている場合に、この複数種別の各分割データとそれぞれの再生時刻情報とを関連付けて生成されるインデックス情報をサーバに要求して受信するインデックス情報受信機能と、
前記分割データの送信をサーバに要求してから、この分割データを受信するまでの時間を推定する時間推定機能と、
当該時間推定機能によって推定された時間に基づいて、要求する分割データの種別を選択する分割データ選択機能と、
前記インデックス情報受信機能によって受信されたインデックス情報にしたがって、再生開始を指示された再生時刻情報及びその後の再生時刻情報に関連付けられ、かつ前記分割データ選択機能によって選択されている種別の分割データを、サーバに順次要求して受信する分割データ受信機能と、
当該分割データ受信機能によって受信された分割データを記憶する記憶機能と、
前記記憶機能によって記憶されている分割データを順次再生させる再生機能とを実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項１０】　請求項９記載のコンピュータ読み取り可能な記録媒体において、
前記分割データ受信機能は、前記時間推定機能によって推定された時間とほぼ同じ時間分の分割データが前記記憶機能に記憶されていない場合に、前記データ選択機能によって選択されている種別の分割データを、前記インデックス情報にしたがって前記サーバに要求し、
前記再生機能は、前記時間推定機能によって推定された時間とほぼ同じ時間分の分割データが前記記憶機能によって記憶された場合に、この記憶機能によって記憶されている分割データの再生を開始することを特徴とするコンピュータ読み取り可能な記録媒体。

【請求項１１】　コンピュータに、
動画像データを複数に分割して複数の分割データを生成し、かつ前記複数の分割データとこの分割データの再生時刻情報とを関連付けたインデックス情報を生成する機能を実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項１２】　コンピュータに、
動画像データが分割されて得られる複数の分割データについて伝送速度が異なる複数の種別を生成し、この複数の種別の各分割データをそれぞれの再生時刻情報に関連付けたインデックス情報を生成する機能を実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体。

【請求項１３】　サーバに記憶されている動画像データをクライアントに配信する動画像データ配信システムにおいて、
前記サーバは、
前記動画像データが分割されて得られる複数の分割データと、この各分割データとそれぞれの再生時刻情報とを関連付けたインデックス情報とを保持し、
前記クライアントは、
前記インデックス情報を前記サーバに要求して受信するインデックス情報受信手段と、
当該インデックス情報受信手段に受信されたインデックス情報にしたがって、再生開始を指示された再生時刻情報及びその後の再生時刻情報に関連付けられた分割データを前記サーバに順次要求して受信する分割データ受信手段と、
当該分割データ受信手段によって受信された分割データを記憶する記憶手段と、
前記記憶手段に記憶されている分割データを順次再生させる再生手段とを備えることを特徴とする動画像データ配信システム。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、サーバの構成及び機能を変更することなく、動画像データのランダムアクセスを可能にするための動画像データ再生装置及び動画像データ分割装置並びにプログラムを記録したコンピュータ読み取り可能な記録媒体に関する。

【０００２】

【従来の技術】従来の動画像データ配信システムは、主にクライアントとＷＷＷサーバとから構成される。この従来の動画像データ配信システムでは、まずクライアントからＷＷＷサーバに再生を求める動画像データ（動画コンテンツ）の伝送要求が送信される。

【０００３】ＷＷＷサーバは各種データを保持しており、このデータの中から要求された動画像データを選択し送信する。

【０００４】送信された動画像データは、伝送要求を送信したクライアントに受信され、ストリーム再生がなされる。

【０００５】このような従来の動画像データ配信システムにおいて、クライアント及びＷＷＷサーバがインターネットを介して接続されている場合には、一般的にＨＴＴＰプロトコルにより動画像データの伝送がなされる。

【０００６】ここで、例えば動画像データの再生を求めるユーザの中には、動画像データの途中位置からの再生（以下、「ランダムアクセス」という）を求める者もいる。

【０００７】従来の動画像データ配信システムによってランダムアクセスを実現するためには、ランダムアクセスを可能とする専用のＨＴＴＰサーバをこの動画像デー

Aylo Exhibit 1002
Page 321 of 1405

タ配信システムに適用する必要がある。

【０００８】また、別の手法として、ランダムアクセスを可能とする専用のプログラムであるＣＧＩ（Common Gateway Interface）をWWWサーバに実装させる必要がある。

【０００９】この専用のＨＴＴＰサーバや、ＣＧＩを実装したWWWサーバは、動画像データを再生するクライアントから独自の要求を受信し、動画像データを指定された位置からＨＴＴＰプロトコルによって送信する機能を持つ。

【００１０】

【発明が解決しようとする課題】しかしながら、上記のように、ランダムアクセスを可能とするために専用のＨＴＴＰサーバを用いたり、サーバにＣＧＩを実装する方法においては、従来の動画像データ配信システムのWWWサーバに手を加える必要があり、現状のインフラのままでは実現することができない。

【００１１】したがって、ランダムアクセスの実現、導入が困難であり、作業が複雑である。

【００１２】また、この従来の方法では、ランダムアクセスのための処理が主にサーバでなされるため、サーバの負荷が増加するという問題が生じる。特に、動画像データ配信システムが複数のクライアントを有する場合には、その傾向が強くなる。

【００１３】本発明は、以上のような実状に鑑みてなされたもので、サーバの構成や機能に手を加えることなく動画像データのランダムアクセスを可能とするための動画像データ再生装置及び動画像データ分割装置並びにプログラムを記録したコンピュータ読み取り可能な記録媒体を提供することを目的とする。

【００１４】

【課題を解決するための手段】本発明の骨子は、動画像データを複数の分割して分割データとし、この分割データにそれぞれの再生時間を示す再生時刻情報を関連づけてインデックス情報とし、このインデックス情報にしたがって分割データを順次再生する点にある。

【００１５】以下、本発明を実現するにあたって講じた具体的手段について説明する。

【００１６】第１の発明は、動画像データが分割されて得られる複数の分割データとそれぞれの再生時刻情報とを関連付けて生成されるインデックス情報をサーバに要求して受信するインデックス情報受信手段と、当該インデックス情報受信手段に受信されたインデックス情報にしたがって、再生開始を指示された再生時刻情報及びその後の再生時刻情報に関連付けられた分割データをサーバに順次要求して受信する分割データ受信手段と、当該分割データ受信手段によって受信された分割データを記憶する記憶手段と、インデックス情報受信手段に受信されたインデックス情報の再生時刻情報にしたがって、記憶手段に記憶されている分割データを順次再生させる再生手段とを具備した動画像データ再生装置である。

【００１７】この第１の発明の動画像データ再生装置においては、再生開始を指示された再生時刻情報以降の分割データが順次受信されストリーム再生される。

【００１８】また、この動画像データ再生装置を適用する場合には、サーバは自己が保持するデータのうち要求されたデータを送信する機能を有していればよい。

【００１９】したがって、第１の発明においては、サーバに新しい構成や機能を付加することなく容易にランダムアクセスが実現される。

【００２０】また、ランダムアクセスを実現させるための処理が、サーバではなく動画像データ再生装置（クライアント）側によって実行されるため、サーバの負荷の増加を防止することができる。

【００２１】第２の発明は、第１の発明の動画像データ再生装置において、分割データの送信をサーバに要求してから、この分割データを受信するまでの時間を推定する時間推定手段を付加し、分割データ受信手段は、当該時間推定手段によって推定された時間とほぼ同じ時間分の分割データが記憶手段に記憶されていない場合に、前記インデックス情報にしたがってサーバに分割データを要求し、再生手段は、時間推定手段によって推定された時間とほぼ同じ時間分の分割データが記憶手段に記憶された場合に、この記憶手段に記憶されている分割データの順次再生を開始する動画像データ再生装置である。

【００２２】この第２の発明の動画像データ再生装置においては、分割データを要求してから受信するまでの間に再生されると推測される分割データが、記憶手段に保持される。

【００２３】したがって、この第２の発明においては、第１の発明と同様の作用効果に加えて、必要最低限の量の分割データが格納されしだい、動画像データの再生がなされるため、再生が開始されるまでの時間を最適化することができる。

【００２４】これにより、再生開始までの時間を短縮させることができる。

【００２５】また、次の分割データが要求されてから受信されるまでの間に必要な分割データは確保済みであるため、動画像データの再生が途切れることを防止することができる。

【００２６】さらに、分割データをサーバに要求してから受信するまでの時間が逐次測定され、この時間にしたがって再生可能な状態で記憶しておく分割データの量が決定される。

【００２７】ゆえに、回線の動的な負荷変動にも対応することができ、適切な量の分割データが確保される。

【００２８】したがって、スムーズなストリーム再生が可能となる。

【００２９】第３の発明は、動画像データが分割されて得られる複数の分割データについて伝送速度が異なる複

Aylo Exhibit 1002
Page 322 of 1405

数の種別が生成されている場合に、この複数種別の各分割データとそれぞれの再生時刻情報とを関連付けて生成されるインデックス情報をサーバに要求して受信するインデックス情報受信手段と、分割データの送信をサーバに要求してから、この分割データを受信するまでの時間を推定する時間推定手段と、当該時間推定手段によって推定された時間に基づいて、要求する分割データの種別を選択する分割データ選択手段と、インデックス情報受信手段に受信されたインデックス情報にしたがって、再生開始を指示された再生時刻情報及びその後の再生時刻情報に関連付けられ、かつ分割データ選択手段によって選択されている種別の分割データをサーバに順次要求して受信する分割データ受信手段と、当該分割データ受信手段によって受信された分割データを記憶する記憶手段と、インデックス情報受信手段に受信されたインデックス情報の再生時刻情報にしたがって、記憶手段に記憶されている分割データを順次再生させる再生手段とを具備した動画像データ再生装置である。

【００３０】この第３の発明の動画像データ再生装置においては、伝送速度の異なる複数種別の分割データのうち、回線の負荷変動を考慮して適した伝送速度の分割データが選択され受信される。

【００３１】したがって、回線の伝送速度が遅い場合には、高速伝送可能な分割データにより伝送がなされるため、再生すべきデータが未伝送となることを防止することができる。

【００３２】また、第１の発明と同様の作用により、サーバを変更することなく、ランダムアクセスが可能である。

【００３３】第４の発明は、第３の発明の動画像データ再生装置において、分割データ受信手段は、時間推定手段に推定された時間とほぼ同じ時間分の分割データが記憶手段に記憶されていない場合に、データ選択手段によって選択されている種別の分割データを、インデックス情報にしたがってサーバに要求し、再生手段は、時間推定手段によって推定された時間とほぼ同じ時間分の分割データが記憶手段に記憶された場合に、この記憶手段に記憶されている分割データの順次再生を開始する動画像データ再生装置である。

【００３４】この第４の発明の動画像データ再生装置においては、第３の発明と同様の作用効果に加えて、分割データをサーバに要求してから受信するまでの間の再生に必要と推測される分の分割データが記憶手段に保持される。

【００３５】したがって、先の第２の発明と同様の効果を得ることができる。

【００３６】第５の発明は、動画像データを複数に分割して複数の分割データを生成し、かつ前記複数の分割データとこの分割データの再生時刻情報とを関連付けたインデックス情報を生成する動画像データ分割装置である。

【００３７】この第５の発明の動画像データ分割装置を用いて複数の分割データ及びインデックス情報を作成してサーバに備え、第１又は第２の発明の動画像データ再生装置を用いることで、サーバの構成、機能に手を加えることなくランダムアクセスが可能となる。

【００３８】また、この動画像データ分割装置を用いることで、複数の分割データ及びインデックス情報を容易に作成できる。

【００３９】第６の発明は、動画像データが分割されて得られる複数の分割データについて伝送速度が異なる複数の種別を生成し、この複数の種別の各分割データをそれぞれの再生時刻情報に関連付けたインデックス情報を生成する動画像データ分割装置である。

【００４０】この第６の発明の動画像データ分割装置を用いて複数種別の各分割データ及びインデックス情報を作成してサーバに備え、第３又は第４の発明の動画像データ再生装置を用いることで、サーバの構成、機能に手を加えることなくランダムアクセスが可能となり、伝送にかかる時間を調節することができる。

【００４１】また、この動画像データ分割装置を用いることで、複数種別の分割データ及びインデックス情報を容易に作成できる。

【００４２】第７乃至第１２の発明は、それぞれ上記第１乃至第６の発明と同様の機能をコンピュータに実現させるためのプログラムを記録したコンピュータ読み取り可能な記録媒体である。

【００４３】この第７乃至第１２の発明のようなプログラムを記録した記録媒体を用いることによって、上述した機能を有していない装置や、計算機においても、簡単に上述した機能を付加することができる。

【００４４】第１３の発明は、サーバに記憶されている動画像データをクライアントに配信する動画像データ配信システムにおいて、サーバは、動画像データが分割されて得られる複数の分割データと、この各分割データとそれぞれの再生時刻情報とを関連付けたインデックス情報とを保持し、クライアントは、インデックス情報をサーバに要求して受信するインデックス情報受信手段と、当該インデックス情報受信手段に受信されたインデックス情報にしたがって、再生開始を指示された再生時刻情報及びその後の再生時刻情報に関連付けられた分割データをサーバに順次要求して受信する分割データ受信手段と、当該分割データ受信手段によって受信された分割データを記憶する記憶手段と、インデックス情報受信手段に受信されたインデックス情報の再生時刻情報にしたがって、記憶手段に記憶されている分割データを順次再生させる再生手段とを備える動画像データ配信システムである。

【００４５】この第１３の発明の動画像データ配信システムにおいては、クライアントが先の第１の発明の動画

Aylo Exhibit 1002
Page 323 of 1405

像データ再生装置と同様に機能する。

【００４６】これにより、第１の発明と同様の作用効果を得ることができる。

【００４７】なお、第２乃至第１２の発明も、この第１３の発明と同様に、動画像データ配信システムに適用可能である。

【００４８】

【発明の実施の形態】以下、図面を参照しながら本発明の実施の形態について説明する。

【００４９】（第１の実施の形態）本実施の形態においては、動画像データ（動画コンテンツ）を複数に分割した分割データと、この複数の分割データとそれぞれの再生時間情報を関連付けたインデックス情報とを用いることで、サーバを変更することなく動画像データのランダムアクセスを可能とする動画像データ配信システムについて説明する。

【００５０】図１は、本実施の形態に係る動画像データ配信システムの構成を示すブロック図である。

【００５１】この動画像データ配信システム１は、動画像データから分割データとインデックス情報とを生成するエンコーダ２と、内部の記憶装置３ａに各種データをファイル形式で記憶するサーバ３と、動画像データ再生用のビューワ４ａを有するクライアント４とから構成されている。

【００５２】この動画像データ配信システム１では、動画像データがスプライスポイントでファイル分割されており、ビューワ４ａがインデックス情報に基づいて好きな位置の分割データをサーバ３に要求し、受信する。

【００５３】以下、この動画像データ配信システム１の各構成要素について詳しく説明する。

【００５４】図２は、本実施の形態に係る動画像データ配信システム１に備えられるエンコーダ２の機能を示すブロック図である。

【００５５】このエンコーダ２は、動画エンコード時に、ビデオ等の動画素材からなる動画像データ（動画コンテンツ）を所定のスプライスポイント毎に分割して複数の分割データを生成する。また生成した各分割データには異なるファイル名を付けて出力する。この図２では、分割データのファイル名の例として「douga000000」、「douga000003」、「douga000006」等が示されている。

【００５６】なお、この各分割データは、ビューワ４ａ内の受信ページバッファの１ページ内に収まるサイズで生成される。

【００５７】さらに、このエンコーダ２は、各分割データのファイル名と、それぞれの再生時刻の情報とを関係付けてインデックス情報５を生成し、ファイル形式で出力する。

【００５８】図３は、本実施の形態に係る動画像データ配信システム１に備えられるサーバ３の構成を示すブロック図である。

【００５９】サーバ３は、主に記憶装置３ａと、サーバ本体３ｂとから構成される。

【００６０】サーバ本体３ｂは、記憶装置３ａに様々なデータを蓄積する機能と、クライアント４からの要求にしたがって記憶装置３ａの内容を読み出して送信する機能を有する。

【００６１】記憶装置３ａは、各種のデータを記憶し、ここではエンコーダ２によって生成された複数の分割データ及びインデックス情報５も記憶している。

【００６２】クライアント４は、図１で示すように、主にクライアント本体４ｂと、動画像データ再生装置として機能するビューワ４ａとから構成されており、クライアント本体４ｂ上でビューワ４ａが稼働する。

【００６３】図４は、本実施の形態に係る動画像データ配信システム１に備えられるビューワ４ａの機能を示すブロック図である。

【００６４】なお、図４（ａ）は再生開始前のバッファリング中の状態を示し、３ページ分の分割データがページバッファに格納されるまでの状態を示している。

【００６５】一方、図４（ｂ）は再生中の状態を示しており、１ページ分の分割データがデコーダに読み出されると、再びサーバ３から分割データが読み出される状態を示している。

【００６６】このビューワ４ａは、動画像データを再生する装置であり、インデックス情報受信部６、分割データ受信部７、ページバッファ８、デコーダ９から構成される。

【００６７】インデックス情報受信部６は、再生を求める動画像データの指定をユーザから入力する。また、指定された動画像データに関するインデックス情報５をサーバ３に要求し、この要求の応答としてサーバ３からインデックス情報５を受信して保持する。

【００６８】分割データ受信部７は、動画像データの再生開始位置となる再生時刻情報をユーザから入力する。また、インデックス情報受信部６のインデックス情報５を参照し、ユーザから入力された再生時刻情報以降の再生時刻情報に関連付けられているファイルの送信をサーバ３に要求する。この要求は、ページバッファ８に格納された分割データの量が一定量（ここでは３つとする）になるまで順次出力する。

【００６９】そして、この分割データ受信部７は、この要求の応答として分割データを順次受信し、ページバッファ８に出力する。

【００７０】ページバッファ８は、複数のページからなり、各ページ毎に一つの分割データをバッファリングする。また、各ページの内容を再生時刻順に並び替えて記憶する。さらに、デコーダ９によって読み出されたページ内の分割データを破棄し、次に入力される分割データを新規に確保する。

Aylo Exhibit 1002
Page 324 of 1405

【００７１】デコーダ９は、ページバッファ８内に一定量の分割データが記憶されると、デコード処理を開始し、その後順次デコード処理を続ける。これにより滞りなく動画像データの再生がなされる。

【００７２】次に、上記のような構成を有する動画像データ配信システム１の動作について以下に説明する。

【００７３】まず、エンコーダ２によって動画像データが分割され、分割データ及びインデックス情報が生成され、サーバ３に記憶される。

【００７４】次に、再生を求める動画像データの指定及び再生を開始させる位置が、ユーザによってビューワ４ａに入力される。

【００７５】次に、ユーザに指定された動画像データから生成されたインデックス情報５が、ビューワ４ａのインデックス情報受信部６からサーバ３に要求され、受信される。

【００７６】この受信されたインデックス情報５には、動画像データの再生時刻情報に対して分割データのファイル名が記述されている。

【００７７】分割データ受信部７では、受信されたインデックス情報５に基づいて、再生を開始したい位置以降に対応付けられている分割データがサーバ３に要求され、受信される。

【００７８】受信された分割データは、ページバッファ８の各ページに格納され、再生される時刻順に並び替えられる。

【００７９】また、ページバッファ８において一定量のバッファリングがなされると、デコーダ９によってストリーム再生が開始される。

【００８０】ページバッファ８においては、分割データが１ページ分再生される度にそのデータが破棄されてページが空けられる。

【００８１】そして、分割データ受信部７において、未受信の次の分割データが、インデックス情報５に基づいて検索され、サーバ３に要求、受信され、再生が繰り返される以上説明したように、本実施の形態に係る動画像データ配信システム１においては、動画像データがスプライスポイントにおいてファイル分割されており、その状態がインデックス情報５に記述されている。

【００８２】また、ビューワ４ａが、このインデックス情報５を要求し、続いてこのインデックス情報５に記されている好きな位置から分割データを順次要求・受信し、再生する。

【００８３】したがって、ユーザの望む位置からの分割データが順次再生されるため、動画像データのランダムアクセスが可能となる。

【００８４】また、本実施の形態においては、サーバ３の機能はファイルを保持し、またこの保持されているファイルの中から要求されたファイルを送信するのみである。したがって、この動画像データ配信システム１を導入する場合にはサーバ３に新しい機能や作用を追加する必要がない。

【００８５】ゆえに、クライアント４側のビューワ４ａを備え、サーバ３に分割データとインデックス情報５とを格納するのみで、容易にランダムアクセスを実現することができる。

【００８６】さらに、ランダムアクセスを実現させるための処理が、サーバ３ではなくビューワ４ａによって実行されるため、クライアント４が複数存在してもサーバ３の負荷の増加を防止することができる。

【００８７】なお、本実施の形態に係る動画像データ配信システム１では、エンコーダ２の機能をクライアント４に実装させ、クライアント４がネットワークを介して分割データ及びインデックス情報５をサーバ３に格納するとしてもよい。

【００８８】また、本実施の形態においては、ビューワ４ａの機能をブラウザ等の機能として実現可能であり、サーバ３はＷＷＷサーバとすることができる。

【００８９】これにより、ブラウザで参照する動画コンテンツのランダムアクセスが可能となる。

【００９０】さらに、本実施の形態に係る動画像データ配信システム１では、クライアント４の台数を複数にすることができる。

【００９１】（第２の実施の形態）本実施の形態においては、第１の実施の形態に係る動画像データ配信システム１に備えられるビューワ４ａを変形したビューワについて説明する。

【００９２】先の第１の実施の形態においては、ビューワ４ａがサーバ３に分割データを要求してから受信するまでの時間がネットワークの回線の負荷に依存する。また、この回線の負荷は、様々な要因により時々刻々と変化する。

【００９３】先のビューワ４ａでは、伝送に要する時間や、回線の負荷による伝送速度の変化を吸収するために、動画像データの再生開始時に一定量分（しきい値を設定する）の分割データをバッファリングし、以後順次分割データを読み出している。

【００９４】この手法では、ビューワ４ａにバッファリングされる分割データの量を大きくすればその分だけ回線の負荷による遅延に対応可能となるが、その分再生開始までの待ち時間が長くなる。

【００９５】この再生開始までの待ち時間は、回線の負荷による遅延を吸収できる範囲で最小化されることが望ましい。したがって、バッファリングされる分割データの量も、回線負荷による伝送時間の変化に対応可能な範囲で最小化されることが望ましい。

【００９６】本実施の形態においては、分割データを要求してから受信するまでの時間を推定する。また、この推定された時間に基づいて、ページバッファに記憶しておく分割データの量を変更し、バッファリングされる分

Aylo Exhibit 1002
Page 325 of 1405

割データの量を動的に最適化するビューワに関して説明する。

【００９７】図５は、本実施の形態に係るビューワの構成を示すブロック図であり、図４と同一の部分には同一の符号を付してその説明を省略し、ここでは異なる部分についてのみ詳しく説明する。

【００９８】このビューワ１０は、第１の実施の形態において説明したビューワ４ａの分割データ受信部７の機能を変更した分割データ受信部１１を備え、新規の構成として時間推定部１２を付加している。また、このビューワ１０は、第１の実施の形態と同様の動画像データ配信システム１に設置可能である。

【００９９】時間推定部１２は、分割データ受信部１１がサーバ３に分割データを要求し、この分割データを受信するまでの時間（以下、「要求受信時間」という）を測定する。

【０１００】なお、この測定された要求受信時間を厳密に用いると、バッファリングの際に余裕がなくなる。したがって、時間推定部１２は多少余分を持たせるために若干多めの値を要求受信時間として推定する。例えば実際の要求受信時間が０．５秒であってとしても、要求受信時間を０．７秒と推定し、設定する。

【０１０１】この時間推定部１２において、要求受信時間は、再生開始以降には分割データが要求されて受信される度に再推定され、更新される。

【０１０２】分割データ受信部１１は、先の第１の実施の形態で説明した分割データ受信部７と同様の処理を行うが、サーバ３から分割データを読み出す際の条件が異なっている。

【０１０３】再生開始前においては、この分割データ受信部１１は、ページバッファ８内の分割データの再生時間の合計が要求受信時間を超えるまで、サーバ３から分割データを受信する。

【０１０４】再生中においては、現在再生中の分割データに関連付けされている再生時刻情報に現状の要求受信時間を加算し、この加算値に該当する再生時刻情報に関連付けされた分割データを要求し、受信する。すなわち、例えば要求受信時間が０．７秒と設定されているとすると、０．７秒後に再生されることになる分割データを要求し、受信する。

【０１０５】上記のような構成を有するビューワ１０の動作について以下に説明する。

【０１０６】図６は、本実施の形態に係るビューワ１０の再生開始前までの動作を示すフローチャートである。

【０１０７】このビューワ１０においては、まずインデックス情報５がサーバ３から受信される（ｓ１）。

【０１０８】次に、このインデックス情報５とユーザの指定内容に基づいて、再生開始位置の分割データが要求される（ｓ２）。

【０１０９】次に、要求された分割データが受信され（ｓ３）、この際に余分を持たせた要求受信時間が推定される（ｓ４）。

【０１１０】次に、インデックス情報５を参照して、次の分割データが要求され（ｓ５）、受信される（ｓ６）。

【０１１１】そして、要求受信時間分の分割データが、ページバッファ８に受信されたか否かが判定され（ｓ７）、受信されている場合には、再生が開始される。

【０１１２】受信されていない場合には、再度次の分割データが要求され、受信される。

【０１１３】図７は、本実施の形態に係るビューワ１０の再生開始後の動作を示すフローチャートである。

【０１１４】再生開始後においては、ページバッファ８に空きがあり（ｔ１）、かつ要求受信時間分の分割データがページバッファ８にない場合に（ｔ２）、次の分割データが要求される（ｔ３）。

【０１１５】なお、この間にもページバッファ８内の分割データはデコードされ再生されている（ｔ４、ｔ９）。

【０１１６】次に、要求された分割データが受信され（ｔ５）、この分割データが最後の分割データか否かが判定される（ｔ６）。

【０１１７】最後の分割データの場合にはデコード処理及び再生が継続され（ｔ７）、処理が終了される。

【０１１８】最後の分割データでない場合には、余分を持たせた要求受信時間が推定され（ｔ８）、デコード処理の実行により動画像データが再生されつつ（ｔ９）、上記の処理が繰り返される。

【０１１９】以上説明したように、本実施の形態に係るビューワ１０においては、サーバに分割データを要求してから受信するまでの時間に多少の余分を加えた時間分の分割データがページバッファ８に蓄積される。

【０１２０】したがって、必要最小限の分割データのみが先行確保されるため、再生開始までの時間を最適化することができる。

【０１２１】（第３の実施の形態）動画像データの再生中に、ネットワークの負荷により回線の伝送速度が分割データの伝送速度未満となると、動画像データがビューワにおいて途切れなく再生されない場合がある。

【０１２２】本実施の形態においては、このような場合に、再生の途中からでも高速伝送可能な分割データに切り換えて伝送を行い、これによりスムーズな再生を維持する動画像データ配信システムについて説明する。

【０１２３】本実施の形態に係る動画像データ配信システムは、図１と同様の構成を有するが、エンコーダ及びビューワに特別の機能が付加されている。

【０１２４】本システムのエンコーダは、元の動画像データが同一であり、スプライスポイントも同一であるが同一の条件における伝送速度が異なる複数種別の分割データ及びインデックス情報を生成する。

Aylo Exhibit 1002
Page 326 of 1405

【０１２５】また、本システムのビューワは、要求受信時間に基づいて分割データの種別を選択する機能を有する。

【０１２６】以下、図１の場合と異なる機能を有するエンコーダ、及びビューワについて詳しく説明する。

【０１２７】図８は、本実施の形態に係る動画像データ配信システムに備えられるエンコーダの機能を示すブロック図である。

【０１２８】このエンコーダ１３は、エンコード時に、１つの動画像データから伝送速度の異なる複数種別の分割データを生成し、出力する。

【０１２９】なお、以下においては伝送速度の異なる２つの種別の分割データが生成される場合を例として説明するが、３つ以上の種別が生成される場合も同様である。また、この２つの種別のうち、一方を標準分割データとし、この標準分割データよりも伝送速度の速いもう一方を高速分割データとして説明する。

【０１３０】さらに、このエンコーダ１３は、生成された標準分割データのファイル名と高速分割データのファイル名を再生時刻情報に関連付けたインデックス情報１４を生成し、ファイル形式で出力する。

【０１３１】すなわち、このインデックス情報１４には、各再生時刻情報に複数列の分割データのファイル名が関連付けられており、この各列は異なる伝送速度になるようにエンコードされた分割データ列である。

【０１３２】サーバは、標準分割データ、高速分割データ、インデックス情報をファイルとして記憶し、クライアントのビューワから要求されたファイルを送信する。

【０１３３】ビューワは、第２の実施の形態に係る図５のビューワ１０と同様の構成を有するが、分割データ受信部１１に分割データ選択機能が付加されている点が異なる。

【０１３４】この分割データ選択機能は、標準分割データを要求してから受信するまでの時間が、時間推定部１２で推定された要求受信時間よりも大幅に大きい場合に、ネットワークの伝送速度が分割データの伝送速度よりも遅いと判断する。この場合、分割データ選択機能は、インデックス情報１４中の分割データの参照を、標準分割データ列から高速分割データ列に切り換える。

【０１３５】一方、高速分割データを要求してから受信するまでの時間が、時間推定部１２で推定された要求受信時間よりもある程度小さくなった場合には、ネットワークの伝送速度が分割データの伝送速度よりも速くなったと判断し、インデックス情報１４中の分割データの参照を、高速分割データから伝送速度は遅いが画質のよい標準分割データ列に切り換える。

【０１３６】図９は、このビューワにおいて再生開始後に実行される処理を示すフローチャートである。

【０１３７】このビューワでは、動画像データがデコードされ、再生されるが（ｕ１）、それと並列に要求伝送時間分の分割データがページバッファ８に格納されているか否かが判定される（ｕ２）。

【０１３８】ここで、要求伝送時間分の分割データがページバッファ８に格納されていない場合には、この要求伝送時間が大きすぎるか否かが分割データ選択機能によって判定される（ｕ３）。

【０１３９】要求伝送時間が大きすぎる場合には、インデックス情報１４の分割データ列のうち伝送速度の速い方が選択される（ｕ４）。

【０１４０】一方、要求伝送時間が大きすぎない場合には、要求伝送時間が小さすぎないか否かが判定される（ｕ５）。

【０１４１】要求伝送時間が小さすぎる場合には、インデックス情報１４の分割データ列のうち伝送速度の遅い方が選択される（ｕ６）。

【０１４２】そして、この分割データ選択機能によって選択された種別における次の分割データが、サーバに要求される（ｕ７）。

【０１４３】この間にも、動画像データのデコード処理が実行され、再生がなされている（ｕ８）。

【０１４４】要求された分割データが受信されると（ｕ９）、余分を持たせた要求受信時間が推定される（ｕ１０）。

【０１４５】そして、要求する分割データがなくなり（ｕ１１）、かつ再生する分割データもなくなると（ｕ１２）、処理が終了する。

【０１４６】要求する分割データが存在する場合には、上記の処理が繰り返される。

【０１４７】以上説明したように、本実施の形態に係る動画像データ配信システムにおいては、異なる伝送速度の分割データが生成され、分割データを要求してから受信するまでの時間が長い場合には、ビューワは伝送速度の速い分割データを読み出す。

【０１４８】また、分割データの種別の切り替えは各分割データの読み出し毎に行うことができるため、動画像データの再生開始後においても伝送速度を変更することができ、ネットワークの状況変更に柔軟に対応することができる。

【０１４９】さらに、伝送速度に余裕のある場合には画質のよい分割データに変更することができる。

【０１５０】ゆえに、クライアントのビューワで動画像データを途切れなく、良好に再生することができる。

【０１５１】なお、上記の各実施の形態において説明したビューワ、エンコーダの機能は、コンピュータに実行させることのできるプログラムとして、例えば磁気ディスク（フロッピーディスク、ハードディスク等）、光ディスク（ＣＤ－ＲＯＭ、ＤＶＤ等）、半導体メモリなどの記録媒体に書き込んで適用したり、通信媒体により伝送して各種装置に適用することも可能である。このビューワやエンコーダの機能を実現するコンピュータは、記

Aylo Exhibit 1002
Page 327 of 1405

録媒体に記録されたプログラムを読み込み、プログラムによって動作が制御されることにより、上述した処理を実行する。

【０１５２】

【発明の効果】以上詳記したように本発明では、動画像データを分解した複数の分割データと、この各分割データとそれぞれの再生時刻情報とを関連付けたインデックス情報をサーバに備える。

【０１５３】そして、動画像データ再生装置は、まずインデックス情報を受信し、再生を開始したい位置の時間再生情報以降に関連付けられている分割データを順次読み出す。

【０１５４】これにより、サーバの構成、機能に手を加えることなく、ランダムアクセスが可能になる。

【０１５５】また、このランダムアクセスのための処理がサーバではなく動画像データ再生装置によって実行されるため、ランダムアクセス実行によるサーバの負荷の増加を防止することができる。

【０１５６】さらに、本発明では、動画像データ再生装置に蓄える分割データの量を、分割データを要求してからこの分割データを受信するまでの時間分とすることで、再生開始までの待ち時間を最適化することができる。

【０１５７】さらに、本発明では、伝送速度の異なる複数種別の分割データを用意しておき、ネットワークの状況によって受信する分割データの種別を切り換えることで、ネットワークにおける負荷の影響を受けることなくスムーズな再生が可能となる。

【図面の簡単な説明】

【図１】本発明の第１の実施の形態に係る動画像データ配信システムの構成を示すブロック図。

＊【図２】同実施の形態におけるエンコーダの機能を示すブロック図。

【図３】同実施の形態に係る動画像データ配信システムに備えられるサーバの構成を示すブロック図。

【図４】同実施の形態に係る動画像データ配信システムに備えられるビューワの機能を示ブロック図。

【図５】本発明の第２の実施の形態に係るビューワの構成を示すブロック図。

【図６】同実施の形態に係るビューワの再生開始前までの動作を示すフローチャート。

【図７】同実施の形態に係るビューワの再生開始後の動作を示すフローチャート。

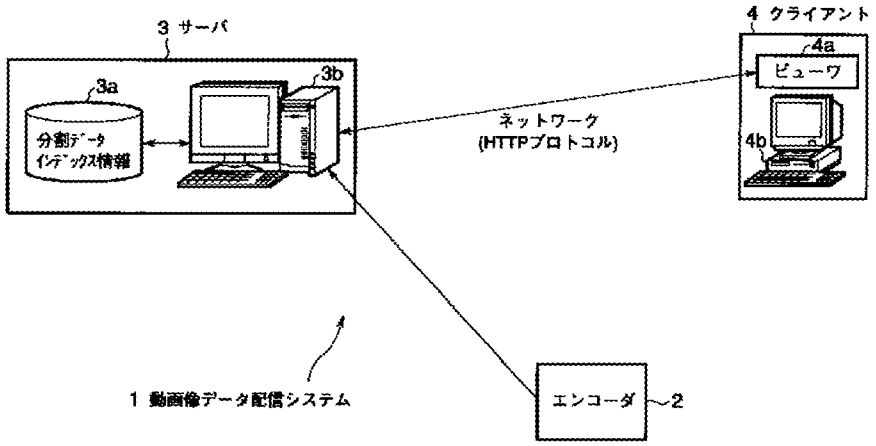
【図８】本発明の第３の実施の形態に係る動画像データ配信システムに備えられるエンコーダの機能を示すブロック図。

【図９】同実施の形態に係る動画像データ配信システムに備えられるビューワの再生開始後の動作を示すフローチャート。

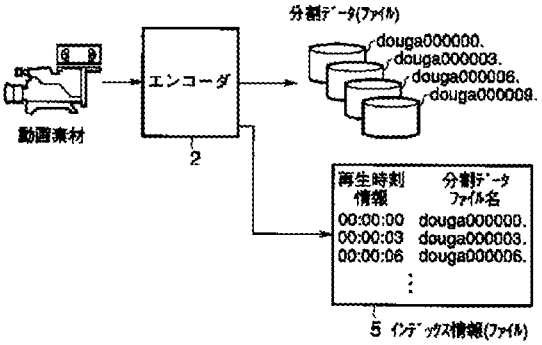
【符号の説明】

１…動画像データ配信システム
２、１３…エンコーダ
３…サーバ
４…クライアント
４ａ、１０…ビューワ
５、１４…インデックス情報
６…インデックス情報受信部
７、１１…分割データ受信部
８…ページバッファ
９…デコーダ
１２…時間推定部

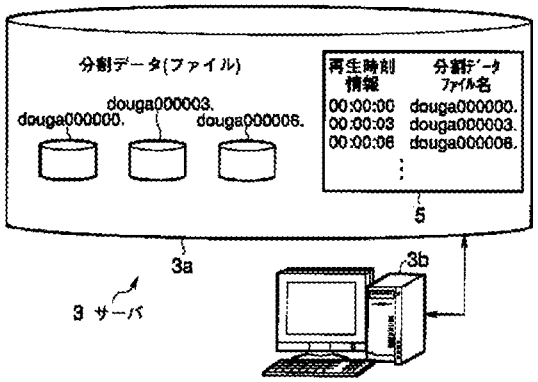
【図１】



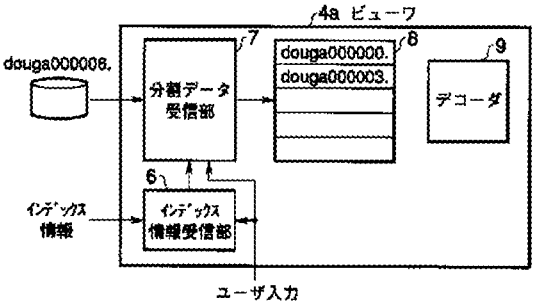
【図２】


分割データ(ファイル)
douga000000.
douga000003.
douga000006.
douga000009.
動画素材
エンコーダ
2
再生時刻情報
分割データファイル名
00:00:00 douga000000.
00:00:03 douga000003.
00:00:06 douga000006.
5 インデックス情報(ファイル)

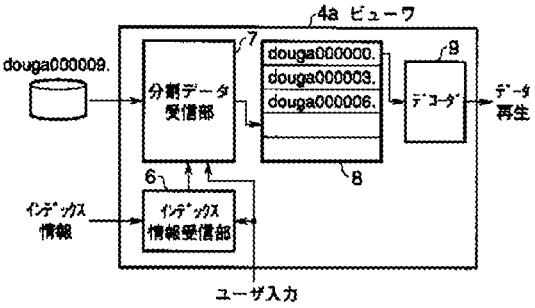
【図３】


分割データ(ファイル)
douga000000.
douga000003.
douga000006.
再生時刻情報
分割データファイル名
00:00:00 douga000000.
00:00:03 douga000003.
00:00:06 douga000006.
5
3a
3b
3 サーバ

【図４】

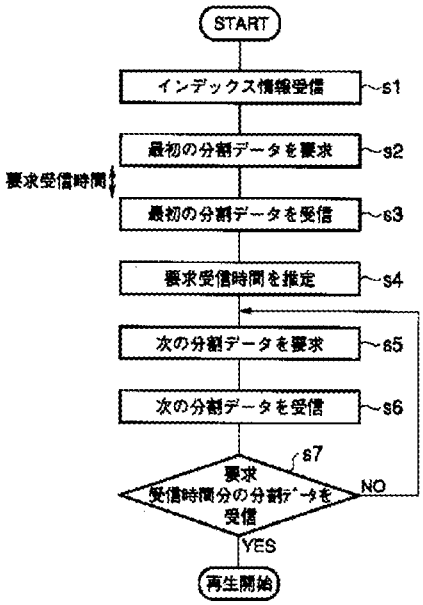

4a ビューワ
douga000006.
分割データ受信部
7
douga000000.
douga000003.
8
デコーダ
9
6
インデックス情報
インデックス情報受信部
ユーザ入力

(a)

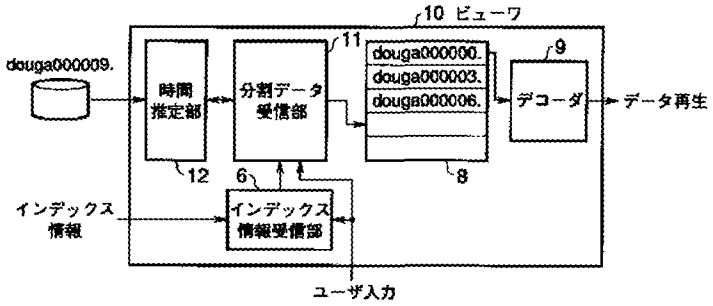

4a ビューワ
douga000009.
分割データ受信部
7
douga000000.
douga000003.
douga000006.
8
デコーダ
9
データ再生
6
インデックス情報
インデックス情報受信部
ユーザ入力

(b)

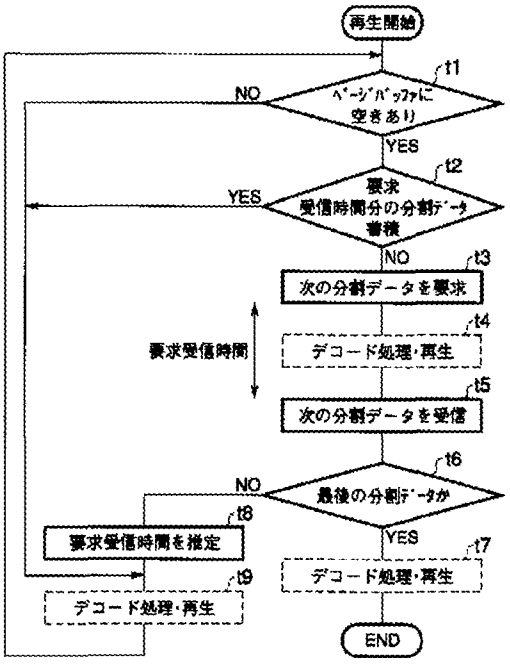
【図６】


START
インデックス情報受信
s1
最初の分割データを要求
s2
要求受信時間
最初の分割データを受信
s3
要求受信時間を推定
s4
次の分割データを要求
s5
次の分割データを受信
s6
s7
要求受信時間分の分割データを受信
NO
YES
再生開始

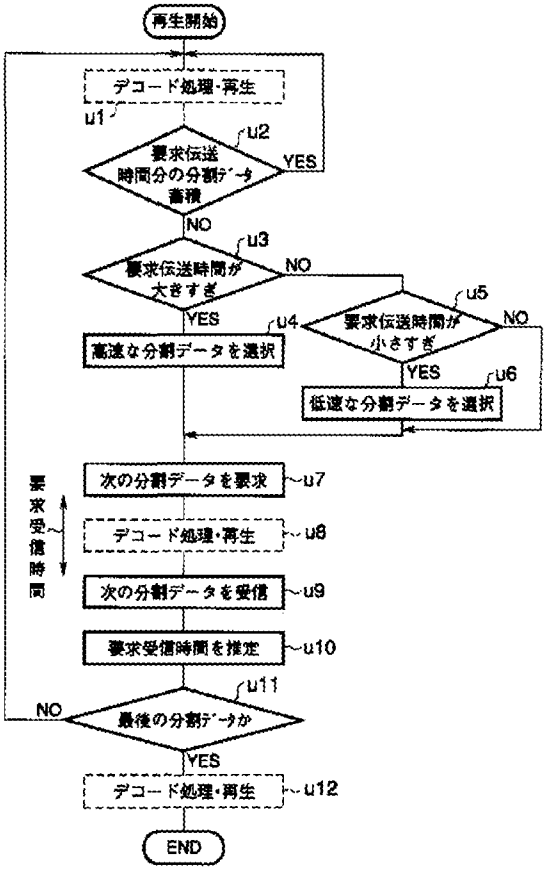
【図５】


10 ビューワ
douga000009.
時間推定部
分割データ受信部
11
douga000000.
douga000003.
douga000006.
9
デコーダ
データ再生
12
6
8
インデックス情報
インデックス情報受信部
ユーザ入力

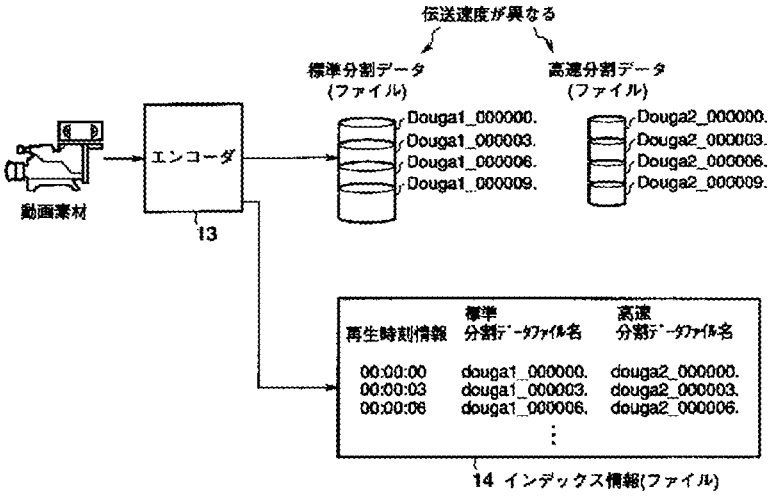
【図７】


再生開始
t1
NO
ページバッファに空きあり
YES
t2
YES
要求受信時間分の分割データ蓄積
NO
t3
次の分割データを要求
要求受信時間
t4
デコード処理・再生
t5
次の分割データを受信
t6
NO
最後の分割データか
YES
t8
要求受信時間を推定
t7
デコード処理・再生
t9
デコード処理・再生
END

【図９】


再生開始
デコード処理・再生
u1
u2
要求伝送時間分の分割データ蓄積
YES
NO
u3
要求伝送時間が大きすぎ
NO
YES
u4
高速な分割データを選択
u5
要求伝送時間が小さすぎ
NO
YES
u6
低速な分割データを選択
要求受信時間
次の分割データを要求
u7
デコード処理・再生
u8
次の分割データを受信
u9
要求受信時間を推定
u10
u11
NO
最後の分割データか
YES
デコード処理・再生
u12
END

【図８】


伝送速度が異なる
標準分割データ
(ファイル)
高速分割データ
(ファイル)
Douga1_000000.
Douga1_000003.
Douga1_000006.
Douga1_000009.
Douga2_000000.
Douga2_000003.
Douga2_000006.
Douga2_000009.
エンコーダ
動画素材
13
再生時刻情報
標準
分割データファイル名
高速
分割データファイル名
00:00:00
00:00:03
00:00:06
douga1_000000.
douga1_000003.
douga1_000006.
douga2_000000.
douga2_000003.
douga2_000006.
14 インデックス情報(ファイル)

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
13 September 2001 (13.09.2001)

PCT

(10) International Publication Number
WO 01/67264 A1

(51) International Patent Classification[7]: G06F 15/16

(21) International Application Number: PCT/US01/40264

(22) International Filing Date: 8 March 2001 (08.03.2001)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
09/520,434 8 March 2000 (08.03.2000) US

(71) Applicant *(for all designated States except US)*: **STREAMCENTER, INC.** [US/US]; 1512 Wellington Place, Aberdeen, NJ 07747 (US).

(72) Inventor; and

(75) Inventor/Applicant *(for US only)*: **MOGHE, Pratyush** [US/US]; 3002 Cricket Circle, Edison, NJ 08820 (US).

(74) Agents: **DIPPERT, William, H.** et al.; Cowan, Liebowitz & Latman, P.C., 1133 Avenue of the Americas, 35th Fl., New York, NY 10036 (US).

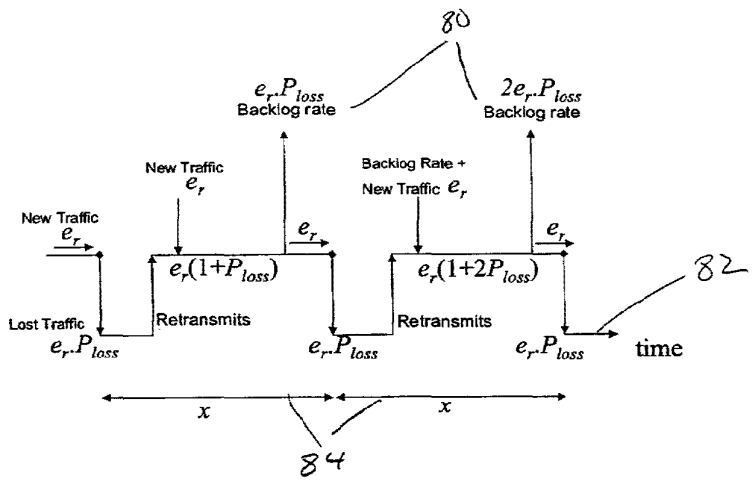
(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, 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, TR), 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 receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(54) Title: APPARATUS AND METHOD FOR PREDICTABLE AND DIFFERENTIATED DELIVERY OF MULTIMEDIA STREAMING ON THE INTERNET



(57) Abstract: The inventive method detects network congestion which interferes with streaming of multimedia audio and video packets from and reacts to it and guarantees delivery of packets. The streaming of the multimedia audio and video packets is performed when a server process executing on a sending computing device sends the packets to a player process executing on a receiving computing device via a network, such as the Internet. The detection of the network congestion is performed by communicating configuration information including network parameters from a service manager via the network to the at least one sending computing device and to the at least one receiving computing device. The network congestion (80) is then confirming by transmitting and retracting network probes. At least one buffer management technique is executed to limit degradation in the streaming of the packets. The end of the network congestion is detected by transmitting and retracting said network probes and the rate of streaming of the packets is configured accordingly (82, 84).

WO 01/67264 A1

APPARATUS AND METHOD FOR PREDICTABLE AND DIFFERENTIATED DELIVERY OF MULTIMEDIA STREAMING ON THE INTERNET

TECHNICAL FIELD

This invention relates to the field of multimedia audio and video streaming on the Internet, and more particularly, to an apparatus and method for predictable and differentiated delivery of multimedia streaming on the Internet.

BACKGROUND OF THE INVENTION

Multimedia streaming involves broadcast of audio and video information from content providers called streaming servers or senders via the Internet to clients or receivers on user computing devices called streaming players. Real-time delivery of the multimedia streaming on the Internet is inherently uncontrollable and consequently unpredictable. When an Internet delivery path is congested, packets of multimedia streaming are typically dropped, thus causing degraded audio and video playback to be received by the streaming players. Typically, manifested forms of congestion can be classified into two types: instantaneous and sustained (longer-term). Instantaneous congestion is caused by random variations in the arrival patterns of packets of multimedia streaming at different Internet routers and switches, and due to the varying sizes of these packets.

These variations cause varying amounts of delay as the packets of multimedia streaming travel from the streaming server to the streaming receiver. If the delay accumulates beyond a certain point, the packets of multimedia streaming can get lost. This type of congestion typically causes sporadic delays and losses of over hundreds of milliseconds to a few seconds, and is hence referred to as "instantaneous congestion." In contrast, sustained congestion is longer-term and results during periods of excessive demand by the streaming players, e.g., demand for multimedia streaming from popular websites over the network paths on the Internet. During these periods, packet traffic surges can overwhelm the packet buffers in network routers, which are network traffic routing and forwarding computing devices, and packets of multimedia streaming can get

lost in bursts. This is similar to a rush-hour traffic scenario on highways at peak hours. Unlike instantaneous congestion, sustained congestion can last from tens of seconds up to a few minutes.

It is well known that instantaneous congestion is an easy problem to solve. The client or the streaming player can buffer incoming packets of multimedia streaming for a short duration before playback. This reduces the effect of short-term delay between packets of multimedia streaming. Additionally, by requesting that a lost packet be immediately retransmitted, the streaming player can counter the loss of packets of multimedia streaming due to instantaneous congestion. Several existing streaming systems use these techniques, i.e., short-term buffering along with retransmits of lost packets, to isolate the streaming player from the effects of instantaneous congestion. Examples of these systems include the Robust-User Datagram Protocol (UDP) feature in the Real system from Real Networks and the NetShow 3.0 MediaPlayer from Microsoft.

The problem of sustained congestion is harder to solve. This difficulty relates to the fact that the Internet delivery lacks a uniform real-time control. Several standardization and research efforts have proposed ways to introduce such control; however, none have succeeded in creating a global Internet standard. The referred to standardization and research efforts include the following:

1. L. Kleinrock, "On Flow Control in Computer Networks", International Conference on Communications (ICC'78), pages 27.2.1-27.2.5;
2. D.E. Comer, R. S. Yavatkar, "A Rate-based Congestion Avoidance and Control Scheme for Packet Switched Networks", The 10th International Conference on Distributed Computing Systems, pages 390-397, 1990;
3. K.K. Ramakrishnan, R. Jain, "A Binary Feedback Scheme for Congestion Avoidance in Computer Networks", ACM Transactions on Computer Systems, Vol. 8, No. 2, May 1990, Pages 158-181; and
4. D. Mitra, "Asympotically Optimal Design of Congestion Control for High Speed Data Networks", IEEE Transactions on Communications, Vol. 40, No. 2, February 1992.

For example, the RSVP effort, described in Braden, R., Zhang, L., Berson, S., Herzog, S., and Jamin, S., "Resource Reservation Protocol (RSVP) - Version 1

Functional Specification", RFC 2205, Proposed Standard, September 1997, seeks to add "quality of service" control within each network router and switch within the Internet. Another effort that seeks to create fast paths on the Internet is the "differentiated service" effort within the Internet Engineering Task Force (IETF) described in the following publications:

1) Nichols, K., Jacobson, V. and Zhang, L., "A Two-bit Differentiated Services Architecture for the Internet", Internet Draft, December 1997;
2) Ferguson, P., "Simple Differential Services: IP TOS and Precedence, Delay Indication, and Drop Preference", Internet Draft, November 1997; and
3) Clark, D. and Wroclawski, J., "An Approach to Service Allocation in the Internet", Internet Draft, July 1997.

The problem with these approaches is that they require structural changes throughout the Internet. This requirement assumes an explicit cooperation between various managers of the equipment comprising the Internet, which is an impractical assumption. Consequently, these efforts have stalled or are expected to have limited success. An alternative approach is to offer "adaptive applications". This approach proposes that application content be adapted based on the level of congestion. Methods based on this approach degrade the "experience" of an application in a controlled manner, but do not attempt to guarantee predictability or isolation from congestion effects. This approach is discussed in I. Busse, B. Deffner, H. Schulzrinne, "Dynamic QoS Control of Multimedia Applications based on RTP", May 30, 1995 and Real Networks, "White Paper on SureStream Technology", Available at http://www.real.com/devzone/library/whitepapers/surestrm.html.

What is needed is a "delivery framework" that will isolate streaming players from the effect of sustained congestion by providing a "predictable" delivery. Furthermore, there is also a need for a "differentiated streaming" framework that will provide for multiple levels of delivery isolation to the streaming players, without requiring structural changes in the Internet, as in K. Nichols et al., P. Ferguson and D. Clark, discussed above.

SUMMARY OF THE INVENTION

The object of the present invention is to achieve a particular level of delivery isolation on particular streaming players. This level of delivery isolation may be based on factors like an importance "value" associated with particular streaming players, or the price the user of the streaming players are charged, etc. As an example, the streaming server may place specific streaming players in the high "value" category based on the demographic information on their owners, while other streaming players may be placed in the low "value" category.

Another object of the present invention is to ensure that the high value users of streaming players always receive consistent, predictable delivery of streaming content, even if there is an excessive user demand. If there is a situation of sustained congestion, wherein not all users can be satisfied, the content provider might want to ensure that the high value users are promised predictable delivery.

Accordingly, the present invention provides a method and an apparatus for predictable and differentiated streaming on the Internet. The system consists of software and random-access memory-based buffers installed at the streaming server and at the streaming players along with a service manager at a separate location. The method prescribes the manner in which the system is used to achieve predictable and differentiated delivery of streaming multimedia packets.

The invention is based on streaming multimedia packets between the software on the streaming server and the software on the streaming. The buffer sizes in the computing devices of the streaming player and of the streaming server are initialized based on the configuration information provided by the service manager software at the start of the execution of the inventive method. This configuration information includes:

a) the expected duration of sustained congestion periods,

b) the worst-case packet loss rate during the sustained congestion period,

c) the encoded bit-rate of streaming source information, and etc.

During the delivery of the streaming multimedia packets, the software on the streaming server relays the streaming packets to the software on the streaming player, which stores the packets in a playout buffer before playing them out. The playout buffer takes care of

Aylo Exhibit 1002
Page 336 of 1405

delay variation that may arise due to instantaneous congestion on the network path. In case of short-term losses caused by instantaneous congestion, the software on streaming player immediately requests the server-end software to retransmit the lost information.

Predictable Delivery Method

To deal with sustained congestion ("congestion") the invention incorporates the following method. The software on the streaming player uses estimation techniques based on probes to detect and confirm the onset of sustained congestion. Once the congestion status is verified, the streaming player software informs the streaming server software about this event. The streaming server software then modifies the manner in which it streams packets to the streaming player software. In particular, the streaming server software ensures that the transmission rate of packets never exceeds the encoded bit-rate of the streaming source. Furthermore, it gives preemptive priority to the retransmitted packets over new packets. This creates a backlog of new packets at one of the streaming server buffers.

Meanwhile, the streaming player software continuously monitors the status of received packets and detects the end of congestion through estimation techniques based on probes. Once the end of congestion is confirmed, the streaming player software informs the streaming server software about this event. The streaming server software then begins to stream its backlog of new packets to the streaming player software. The transmission rate of this stream is increased based on the estimation information from the streaming player software along with feedback obtained from other streaming players in the network. This rate is carefully selected to ensure that the network is not flooded back into congestion. The buffer sizing at the streaming player and at the streaming server is based on the configuration information and ensures that the user experiences an uninterrupted playback during sustained congestion. This assumes that the configuration information is the "worst-case" scenario. If the actual service parameters turn out to be worse than the configuration information, the streaming player software contacts the service manager after the service. The configuration information at the service manager is now updated in a persistent manner with the actual worst-case parameters.

In one preferred embodiment, this update happens as follows. At the next service invocation, the buffers are sized based on the updated configuration information. The method thus automatically learns the worst-case behavior from the actual behavior of the network each time. Eventually, the method can guarantee a predictable delivery based on a non-interrupted playout in most cases. Various other embodiments are also possible, wherein the buffer size is fixed, but the encoding bit-rate of the stream is modified to choose the largest encoding bit-rate that leads to an uninterrupted playback.

Differentiated Streaming Method

The method for differentiated streaming involves identical operations as the predictable streaming method, except for the mechanism for streaming the backlog of packets from the streaming server software. For differentiated streaming, the streaming server software transmits the stream based on the "value" associated with the particular user, in addition to the information used for predictable streaming.

Accounting Delivery Service

Additionally, the methodology of the mentioned invention is utilized to offer the inventive method as an "accountable delivery service" to a large number of content providers, i.e., streaming servers, and users, i.e., streaming players. Such a service may deliver performance contract-based services to users of the streaming players on the Internet.

The service manager plays a central role in this service, by keeping and exchanging the state of several streaming players, delivery paths, and content providers/streaming servers in its database. The method of the present invention tracks the "actual" delivered viewing/listening experience of users of streaming servers by computing the delivery statistics including but not restricted to metrics that quantify "pauses or interruptions", "content throughput", and "content loss". In addition to the features of predictability and differentiation, the delivery service can also answer the question of "service feasibility". That is, at the beginning of a session, the service manager can estimate, based on the historical and instantaneous information, whether a

particular user can receive an "expected quality experience." While the present invention does not describe the details of this service, it should be obvious to those skilled in the art to build such a service around the framework of the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and other objects and advantages of the invention will be apparent upon consideration of the following detailed description, taken in conjunction with the accompanying drawings, in which the reference characters refer to like parts throughout and in which:

Figure 1 is a schematic diagram of the devices and programs comprising the invention.

Figure 2 is a schematic diagram of the components of the devices utilized by the invention.

Figure 3 is a schematic diagram of the operation of the invention during normal streaming.

Figure 4 is a schematic diagram of the operation of the invention during congestion.

Figure 5 is a schematic diagram of the recovery scheduling operation of the invention after congestion.

Figure 6 is a diagram of the modeling the backlog operation of the invention during congestion.

Figure 7 is a state transition diagram for streaming player module (normal streaming).

Figure 8 is a state transition diagram for streaming server module (normal streaming).

Figure 9 is diagram of a congestion processing protocol at the streaming player module (during and after congestion).

Figure 10 is a diagram of an congestion processing protocol at the streaming server module (during and after congestion).

DETAILED DESCRIPTION OF THE INVENTION

Architecture

The invention, shown in Figure 1, comprises at least one computing device 10 for executing a service manager process 12, at least one computing device 14 for executing streaming server process 16, and at least one computing device 18 for executing streaming player process 20. The service manager process 12 is shown as residing on the computing device 12 only for the clarity of presentation. It may be obvious to those skilled in the art that the service manager process 12 may reside on any computing devices 14 and 18. Furthermore, server processes 16 may be present on the computing devices 18 alongside player processes 20, and vice versa the player processes 20 may be present on the computing devices 14 alongside server processes 16. Each computing device 10, 14 and 18 are connected to a network 22 and establish data paths amongst each other. In the preferred embodiment, the network is the Internet.

The streaming server process 16 transmits a single multimedia stream to each streaming player process 20 and the streaming player process 20 receives that single multimedia stream. The streaming server process consists of three buffers:

a) Output buffer 24, used to transmit new packets from the server process 16 to the player process 20;

b) Backup buffer 26 , for maintaining temporary copies of the packets transmitted to the player process 20; and

c) Retransmit buffer 28, for maintaining retransmit requests received from the player process 20. The streaming player process 20 may comprise a player buffer 29.

Before a streaming session begins, the Service Manager process 12 downloads configuration information via the network 22 to both the computing device 14 for the use of the streaming server process 16 and to the computing device 18 for the use of the streaming player process 20. At the end of each streaming session, the configuration information on the computing device 10 is updated for the future use of the Service Manager process 12 via the network 22 by both the streaming server process 16 on the computing device 14 and the streaming player process 20 on the computing device 18.

The computing devices 10, 14, and 18 may take the configuration of any computer ranging from mainframes and personal computers (PCs) to digital telephones and hand held devices, e.g., palm pilots™. In one illustrative embodiment of this invention shown in Figure 2, such computing devices may comprise a bus 30, which is connected directly to each of the following:

1. a central processing unit (CPU) 32;
2. a memory 34;
3. a system clock 36;
4. a peripheral interface 38;
5. a video interface 40;
6. an input/output (I/O) interface 42;
7. a communications interface 44; and
8. a multimedia interface 46.

The common bus 30 is further connected

1. by the video interface 40 to a display 50;
2. by the I/O interface 42 to a storage device 52, which may illustratively take the form of memory gates, disks, diskettes, compact disks (CD), digital video disks (DVD), etc.;
3. by the multimedia interface 46 to any multimedia component 56;
4. by peripheral interface 38 to the peripherals 58, such as the keyboard, the mouse, navigational buttons, e.g., on a digital phone, a touch screen, and/or writing screen on full size and hand held devices, e.g., a palm pilot ™;
5. by the communications interface 44, e.g., a plurality of modems, to a network connection 60, e.g., an Internet Service Provider (ISP) and to other services, which is in turn connected to the network 22, whereby a data path is provided between the network 22 and the computing devices 10, 14, and 18 (Figure 1) and, in particular, the common bus 30 of these computing devices; and
6. furthermore, by the communications interface 44 to the wired and/or the wireless telephone system 54.

Normal Conditions

Turning now to Figure 3, the normal streaming operation is shown when there are either zero, i.e., no losses, or sporadic streaming losses, i.e., loss occurring due to instantaneous burst of streaming packets. The overall streaming sequence works as follows. Packets streamed from the streaming server process 16 are queued up in the streaming server output buffer 24 for transmission. The server process 16 is responsible for transmitting streaming packets 70 to the computing device 18 executing the player process 20. Periodically, this process 16 checks the output buffer 24 and the retransmit buffer 28 to determine if there are streaming packets 70 awaiting transmission or any requests form the player process. Any remaining streaming packets in the output buffer 24 are then transmitted. If retransmission requests are found, any streaming packets 70 requested for retransmission and found in the backup buffer 26 are transmitted. When streaming packets 70 are transmitted, a temporary backup copy of the transmitted streaming packets 70 is kept in the backup buffer.

The streaming player process 20 receives the streaming packets 70 and stores them at the tail-end of the player buffer 29. Periodically after a user-defined interval, the player process 20 examines the received streaming packets 70 to see if any streaming packets 70 are missing. If packets are missing, the player process 20 sends a request 72 to the server process 16 to retransmit the missing packets. After assuring that all the streaming packets 70 are received, the received streaming packets 70 are forwarded from the head-end of the player buffer to the streaming player process 20.

Congested Conditions

The invention detects congestion early on, and then may take steps so that the impact of congestion on the stream is minimized. A correctly configured framework can keep a stream completely transparent of sustained congestion, thus providing an effect equivalent to a "reservation." Two principles are at the basis of this framework, detection/reaction to congestion and congestion recovery.

1. Detecting and reacting to congestion

The invention detects congestion through probes launched by the player process 20. The probes are discussed in detail later. Once congestion is confirmed, the server process 16 reacts by using a technique called "constant bit-rate emulation." This name derives from the fact that the server process 16 tries to maintain a constant upper bound on the rate at which the streaming packets 70 are streamed to the player process 20. Typically, the upper bound is set equal to the encoding bit rate of the stream. In effect, the server process 16 emulates a constant bit-rate by substituting new server packets with packets that have been requested 72 for retransmission. Please recall that in congestion, the server has a large number of retransmit requests 72 issued from the player process 20. Figure 4 shows the flow of streaming packets 70 during this congestion situation. Two important reasons motivate the constant bit-rate emulation method:

a) The method demonstrates a "network-friendly" yet efficient behavior. The method is network-friendly, since it conserves bandwidth during congestion by lowering the upper bound of its streaming transmission rate. Additionally, the method is efficient for the following reasons. First, the upper bound on the transmission rate is set equal to the maximum encoding bit-rate of the source stream; this is not a restrictive setting. Unlike other schemes that voluntarily reduce the source rate during congestion, the present invention only reduces the transmission rate by the amount of incurred loss. This decision has an important implication in practice. Most practical congestion-processing algorithms will incur errors and delays in estimating and reacting to the start of congestion or end of congestion. Consequently, there will be temporary periods in which these algorithms "think" that the network is congested, when in practice, the network has cleared out of congestion. In situations like this, estimation errors do not degrade the streaming throughput. In fact, the invention delivers a throughput better or equal to the throughput of any other scheme, if there is no loss of streaming packets 70.

b) Assuming most connections sharing the network with streaming are using the Transmission Control Protocol (TCP), the constant bit-rate of the invention in congestion allows the TCP traffic on the same path to accurately sense the available bandwidth (most Internet or Web traffic uses TCP). An alternative design with

adapting bit-rate would confuse most other TCP connections and cause each to keep oscillating their TCP windows. If all TCP connections maintain a steady window-size, the performance of the network is much smoother in congestion. This behavior can drain out the congestion faster and reduce the estimation errors.

2. Detecting End-of-Congestion and Recovery from congestion

The invention detects the end of congestion through probes launched by the player process 20. Once the player process 20 confirms that congestion is over, the server process 16 enters a phase called "recovery scheduling". Figure 4 shows a recovery scheduling after congestion. To understand this phase, recall that in the previous state of congestion, the server transmits packets at a constant rate. By substituting new packets with older lost packets, the server is in effect creating a backlog of new packets in its output buffer. The idea in recovery scheduling is to exhaust the substantial backlog that may build up at the server output queue. Clearly, this backlog must not be cleared by flooding the network. Such action may cause significant losses in the intermediate switch buffers. In addition, if thousands of streams of packets 70 flood the backlog at the same time, this may cause certain network points to get congested again. Recovery scheduling is a procedure that configures the rate of streaming the backlog based on the network parameters. Additionally, recovery scheduling can also take into account the "premium" placed on the particular player process 20. The latter consideration allows the server process 16 to stream backlog of streaming packets at different priorities. "Higher value" users can receive backlog sooner than "lower value" users. This is the basic principle behind differentiated streaming.

Buffer Design

As described previously, the configuration information is downloaded to the computing devices 14 and 16 at the beginning of each streaming session. This configuration information includes the following parameters:

(1) Worst-case duration of congestion, $c*x$, where x is a sampling or retransmission interval, and c is a number of sampling intervals in a congestion period.

(2) Worst-case of loss of streaming packets 70 during congestion defined by P_{loss}

(3) Encoding bit-rate of the stream of streaming packets 70 defined by e_r

(4) Averaged and minimum service rate, SR_{avg} and SR_{min}. This is the rate at which streaming packets 70 are actually transferred from the server process 16 to the player process 20. SR_{min} is the minimum rate across the path via the Internet 22 from the server process 16 to the player process 20.

(5) Path bottleneck link bandwidth B_{min} and the hop link capacities, where B_{min} is the minimum physical bandwidth across all the hops, i.e., the routing devices between the server process 16 and the player process 20.

The parameters indicated in the list above may be obtained through historical measurements of the relevant values and through measurements made on streaming packets 70 at the computing device 14 of the server process 16. The parameters are in turn used to design the sizes of the buffers 24, 26, 28 and 29 of server process 16 and the player process 20. One preferred embodiment of achieving this is described immediately below. That embodiment ignores the upper bounds limits on the buffer sizes related to the available random-access memory 34 (Figure 2) in the computing devices 14 and 18.

Player Buffer-size (b_c)

The buffer on the computing device 18 used by the streaming player process 20 plays out the streaming packets 70 at the maximum rate of e_r bits per second. During sustained congestion, the packet loss is approximately P_{loss}, and the maximum transmission rate from the server, as explained above is e_r. It follows that the rate of the player buffer 29 fill rate equals to $e_r * (1 - P_{loss})$ bits per second. Thus, the net drain on the player buffer 29 proceeds at the rate of $e_r * P_{loss}$ bits per second. To prevent the player buffer 29 from draining out completely within the congestion period $c * x$, the buffer-size of the player buffer 29 should satisfy the constraint defined as:

$$\frac{b_c}{e_r * P_{loss}} \geq c * x \quad \text{or} \quad b_c \geq c * x * e_r * P_{loss} \qquad \text{(Equation 1)}$$

Streaming Server Output Buffer-size (b_o)

Recall that the streaming server output buffer holds the backlog of new packets created during the congestion period. Consequently, the size of this buffer is dictated by the worst-case amount of backlog during the congestion period. To calculate the worst-case backlog of the streaming packets 70, a simple model may be built up. Figure 6 shows the approximate amount of backlog as a function of time 82 during sustained congestion. As indicated, the backlog rate 80 after two retransmission intervals 84 is $2 * e_r * P_{loss}$. By extension, after c retransmission intervals 84, the backlog rate 80 will be $cx * e_r * P_{loss}$. This establishes the constraint on the size of the output buffer 14 (Figure 1) in the memory 34 (Figure 2) of the computing device 14 (Figure 1) executing streaming server process 16 (Figure 1).

$$b_o \geq c * x * e_r * P_{loss} \qquad \text{(Equation 2)}$$

Server Backup Buffer-size (b_{backup})

The backup buffer 26 (Figure 1) holds temporary copies of the server packets that have been transmitted to the player process 20 (Figure 1). The copies must be held until the server is sure that the packets have been successfully received in the player buffer 29 (Figure 1). To estimate the size of this backup buffer 26 (Figure 1) again, an operational model is built. First, the transmission rate in packets per seconds is defined as R_p. At the start of the first retransmit 84, the approximate rate of loss of the streaming packets 70 (Figure 3) is $P_{loss} * R_p$. Assuming the loss affects new as well as retransmitted streaming packets 70 (Figure 3) equally, the approximate rate of loss for older unsuccessful streaming packets 70 (Figure 3) at the second retransmit 84 is $P_{loss}^2 * R_p$. Supposing now that the number of retransmits that will reduce the loss rate for older

packets to a negligibly small number, e.g., 0.1 packets for every retransmission interval is defined by m. This implies that:

$$P_{loss}^{m} * R_p \leq \frac{0.1}{x}, \quad \text{or that} \quad m \geq \frac{\log\left(0.1/R_p * x\right)}{\log P_{loss}}$$

Alternatively, if s is the packet size, the above equation may be expressed as:

$$m \geq \frac{\log\left(0.1 * s/e_r * x\right)}{\log P_{loss}}$$

The minimum required buffer size may be defined as equal to $m * x$. Consequently, the size constraint of the backup buffer 26 (Figure 1) is

$$b_{backup} \geq m * x \quad \geq x * \frac{\log\left(0.1 * s/e_r * x\right)}{\log P_{loss}} \qquad \text{(Equation 3)}$$

The detailed operational method underlying the invention is captured through a protocol called "Streaming Congestion Protocol" (SCP). In the section below the SCP sequences during different phases of streaming are described. Figures 6 and 7 show the detailed operational sequence at the streaming server process 16 (Figure 3) and the streaming player process 20 (Figure 3) during normal streaming of the streaming packets 70 (Figure 3). The operational sequences are described with the help of state diagrams. A state diagram shows various states and transitions between states. Each state captures a specific operational state, and the transitions between states capture the events that cause the operational states to change. Figure 7 shows the state diagram of events at the streaming server process 16 (Figure 1). Figure 8 shows the corresponding state diagram at the streaming player process 20.

Processing during and after congestion

Please refer to Figure 9, by default, the streaming player process 20 is in a normal state 100. The player process 20 may keep a track of the packet loss of a received stream at a periodic interval that may be user determined. If the amount of loss exceeds a

threshold, e.g., Threshold1, across a time-window of some number of seconds which may also be user determined, e.g., TW1, the player process 20 moves into the Likely Congestion state 102. At this point, the streaming player process 16 (Figure 1) launches congestion probes to determine whether the congestion symptoms are verified. These probes form a part of a test called confirm congestion test. If the test indicates that congestion is confirmed, the process moves to a Congestion Confirmed state 104. Immediately, the streaming player process 20 contacts its server counterpart and informs it about the new state change along with some other relevant information.

Note that the transition from a Likely Congested state 102 to Congestion Confirmed State 104 can happen in two other ways. First, the player process 20 may actually receive a Remote Congestion Confirmed report in state 106 from its server process 16 (Figure 1) indicating that some other player process 20 on a related network path is in the Congestion Confirmed state. If this happens, the player immediately moves to the Congestion Confirmed State 104 itself. If the player process 20 has already started conducting congestion probes before this notification, the player process 20 may halt them at the instant it receives the notification. Alternatively, the player process 20 may receive a Local Congestion Confirmed report in state 102 reporting that some other local player process 20 on the same streaming player module has verified congestion. If the Remote or local Congestion Confirmed reports have verified congestion within a past time-window not exceeding the time defined as Tcc, the player process 20 moves to a confirmed congestion state 104.

Once the streaming player process 20 (Figure 1) enters the congestion confirmed state 104, it periodically monitors the loss of the received stream of streaming packets 70 (Figure 3). If the packet loss is less than a second user defined threshold, e.g., Threshold2, across a second user defined time-window, e.g., TW2 seconds, the player process 20 enters the likely end-of-congestion (EOC) state 106. At this point, the player process 20 launches an EOC probe test that either confirms or invalidates the EOC decision. If the EOC probe test confirms congestion, the streaming player process 20 enters the EOC state 108 and sends an EOC report 110 to its server process 16 (Figure 1) counterpart. With this transition, the process re-enters the normal state 100. Note that

during transitions from the EOC state 110 back to the Normal State 100, the streaming player process 20 also needs to continually process the arriving packets in step 120 (Figure 7), check and request for retransmission of missing packets in step 122 (Figure 7), and forward packets to the streaming player for playback in step 124 (Figure 7). These activities are indicated in Figure 7 and are not shown in Figure 9 for purposes of clarity.

Figure 10 shows the SCP state-diagram of the server process 16. After initialization, the server process 16 enters the normal state 130. If the server process 16 receives a report from its streaming player process 20 (Figure 1) indicating congestion, the server moves to the Confirmed Congestion or Confirmed Congestion State 132. At this point, the server process 16 shares the Confirmed Congestion status information with other server processes 16 running on the same computing device 14 (Figure 1) in the Propagate Confirmed Congestion state 134, and potentially also from other computing device 14 (Figure 1) through interaction with the service manager 12 (Figure 1) via the network 22 (Figure 1). The idea behind this is to have selected server processes report the congestion to their player process so that these player processes can avoid doing the Confirmed Congestion probing test. This cuts down on the congestion reaction time and significantly cuts down the probing traffic. In general, each server process that receives the Confirmed Congestion status in the Receive the Confirmed Congestion status state 140 from other server processes 16, first determines whether the Confirmed Congestion status is relevant to its target player process 20 (Figure 1). If this is the case, the server processes notify their player process 20 (Figure 1). Notification is done through a special signaling scheme involving reliable transport protocol (RTP) messages, as discussed in Schulzrinne, Casner, Frederick, Jacobson, "RTP: A Transport Protocol for Real-time Applications", RFC 1889, Internet Engineering Task Force.

The streaming server process 16 will then make transitions from the Confirmed Congestion State 132. Starting with the Confirmed Congestion state 132, the server process 16 transmits streaming packets 70 (Figure 3) at a constant bit-rate. The server process 16 moves to the Recovery Scheduling State 136 as soon as it receives an End-of-congestion or EOC report from the player process 20 (Figure 1) targeted for the reception of

its content. Thereafter the server process 16 changes its mode of transmission, using recovery scheduling to drain its backlogged output buffer 24 (Figure 1) in an intelligent way. As soon as the backlog is over, the server process 16 moves back to the normal state 130.

While in the Normal State 130 the server process 16 may receive and filter information from other server processes 16 by entering the Receive Confirmed Congestion State 140. It may also notify the player processes 20 (Figure 1) about the confirmed congestion status through the in-band RTP signaling, by entering the Confirmed Congestion Notification state 138 from where the process 16 will return to the Normal state 130.

The server process 16 also needs to continually process the departing packets in step 150 (Figure 8), check and process the requests for retransmission of missing packets in step 152 (Figure 8), and backup packets to the backup buffer 26 (Figure 1) for playback. These activities are indicated in Figure 8 and are not shown in Figure 10 for purposes of clarity.

Recovery scheduling algorithm

Supposing that the server process 16 (Figure 1) streams of streaming packets 70 (Figure 3) are classified into K number of priority classes. Class 1 being the highest priority and class "K" being the lowest priority. Each priority class "j" is associated with a weighting function α^j, where $0 < \alpha^K < \alpha^{K-1} < .. \alpha^j < .. < \alpha^1 = 1$. Considering the server process 16 (Figure 1) with priority equaling to "j", the rule for transmission rate e_t of this process is:

$$e_t = \min(\alpha^j \times SR_{min}, \alpha^j \times \frac{B - \lambda_{other}}{\sum_{i=1}^{i=K} \alpha^i n^i})$$

(Equation 4)

In equation 4, all the terms are measured at the service bottleneck hop between the computing device 14 executing the server process 16 and the computing device 18 executing the player process 16. B is the bottleneck link capacity at the bottleneck hop.

λ_{other} is the aggregate packet rate carried at the hop from traffic not utilizing the inventive method. This rate can be estimated by obtaining the aggregate packet rate of the streaming packets 70 (Figure 3) of the inventive method λ_{SME} from the service manager, and using the fact that $\lambda_{other} = B \times \rho - \lambda_{SME}$, where $\rho = 1 - \frac{SR_{\min}}{B}$ is the utilization of the bottleneck hop. The parameter n^i is defined as the number of priority "i" streams carried at the bottleneck hop. The hop is a section of a network between two network-computing devices such as routers. This parameter may be obtained from the service manager 12 (Figure 1). The logic behind this equation is as follows. The first term indicates that the new transmission rate should be set equal to the minimum service rate along the path, in order to recover from the backlog. The second term takes into account the result of increasing the transmission rate on the utilization at the bottleneck hop. In other words, since the maximum utilization can be 100%, each computing device 14 executing server process 16 must carefully weigh the effects of increasing the transmission rate. The best rate is the maximum rate that does not congest the bottleneck bandwidth during recovery. Hence the min operator. Finally, the priorities within the equation 4 provide for differentiated streaming.

Congestion Estimators

Earlier on, we described the congestion estimation tests as part of the SCP algorithm. In principle, congestion can be estimated in a variety of ways, and the invention is not restricted to any one particular method of congestion estimation. A particular embodiment of the invention for estimating congestion based on a mix of passive and active probes will now be described. This method comprises a set of rules configured around the results of the probes. A specific congestion test e.g., an end of congestion test or a confirmed congestion test mentioned in earlier sections, combines these rules in a particular way.

The following are some of a variety of methods that may be used to compute the bottleneck bandwidth on a path between the streaming player 20 (Figure 1) and the

streaming server 16 (Figure 1). These methods are shown here for exemplary purposes only and not to indicate that only these methods must be used with the present invention.

Capacity Probe

This method is described in detail in Van Jacobson, "pathchar - A tool to infer characteristics of Internet paths", MSRI, April 21, 1997; and R. L. Carter and M.E. Crovella, "Measuring Bottleneck Link Speed in Packet-switched Networks", TR-96-006, Boston University Computer Science Department, March 15, 1996. This method is initiated by the player process 20 (Figure 1) at various stages of a streaming session. The steps of this exemplary method are as follows:

1. Do a traceroute and get hop identities marked by IP addresses.
2.

a) For every hop from one onwards, send a series of ICMP timestamp requests (type 13) from the player to the hop. Packet sizes of queries are changed in steps of 1024, 512, 256, 128, and 64, where each queries are sent out with a spacing of 3 seconds between them.

b) For every hop i between the player and the server, the player must record the ICMP timestamp response (type 14) and compute the time difference between the hop i transmit time t_i and the player receive instant t_1.

c) Denote the time difference by $\tau_{i1} = t_i - t_1$.

d) Take four measurements for timestamp differences by varying packet size through five values. Discard the minimum and the maximum, and take average of the remaining two. Let s' and s'' denote the packet sizes of two selected queries to hop i. Let $\tau_{i1'}$ and $\tau_{i1''}$ denote the measurements for the respective packet sizes.

e) Calculate $X_{i1} = \frac{\tau_{i1'} - \tau_{i1''}}{s' - s''}$

$$B_{i,i-1} = \frac{1}{X_{i1} - \left(\sum_{k=1}^{i-2} \frac{1}{B_{i-k,i-k-1}} \right)}$$

It can be proved that link capacity can be computed recursively.

The path bottleneck bandwidth $B_{min} = \min_i B_{i,i-1}$. (Equation 5)

Service Rate Probe

The "Service Rate probe" computes the approximate service rate of transferring packets between the server and the player, as discussed in Van Jacobson, "pathchar - A tool to infer characteristics of Internet paths", MSRI, April 21, 1997; and R. L. Carter and M.E. Crovella, "Measuring Bottleneck Link Speed in Packet-switched Networks", TR-96-006, Boston University Computer Science Department, March 15, 1996. The minimum service rate is the least service rate on the path, while the average service rate is the "expected" service rate on the path. . The steps of this exemplary method are as follows:

1. Obtain the hop list through traceroute. Suppose there are n-1 hops between the player and the server.
2. Periodically, the player initiates a cycle. In each cycle, the player sends n-1 queries, each spaced by ¼ second. The player forms timestamp differences between query responses from consecutive hops. The results are sent back to the server.
3. The server uses results from several cycles along with the link capacities to calculate the service rate.
   a) First, the server calculates $\tau_{m,m-1}^{min} = min(\tau_{m,1} - \tau_{m-1,1})$ where the min operator works over all collected cycle results. We approximate $\tau_{m,m-1}^{min}$ to be the base-line delay between node m and node m-1.
   b) Denote the maximum queuing delay between node m and node m-1 by $q_{m,m-1}^{max}$, where node is a computing device, e.g., a computer or a router connected to the network.

c) Then, $q_{m,m-1}^{max} = max\left(\tau_{m,m-1} - \tau_{m,m-1}^{min}\right)$

d) Denote the average queuing delay between node m and node m-1 by $q_{m,m-1}^{avg}$.

e) Then, $q_{m,m-1}^{avg} = avg\left(\tau_{m,m-1} - \tau_{m,m-1}^{min}\right)$

Finally, the minimum service rate is:

$$SR_{m,m-1}^{min} = \frac{s}{\frac{s}{B_{m,m-1}} + q_{m,m-1}^{max}},$$

and the average service rate is:

$$SR_{m,m-1}^{avg} = \frac{s}{\frac{s}{B_{m,m-1}} + q_{m,m-1}^{avg}} \qquad \text{(Equation 6)}$$

Congestion Detection Rule (CC Test)

A robust but responsive congestion detection mechanism is an essential part of the inventive method. It is important to respond quickly to congestion; this can reduce the congestion period and hence make it easier for the recovery scheduling to do its job in a shorter period of time. Ultimately this can allow the method to work with a reduced buffer size and hence a reduced connect lag. It is equally important to robustly distinguish a sustained congestion condition. False detection can lead to reduced throughput and performance. Recall that the inventive method treats sustained congestion by reducing the throughput of the newer packets, so that the throughput of older retransmitted packets is maximized.

The present invention uses the following congestion detection rule:

1. Packet losses exceeding a threshold for a consecutive time-window of duration T_s. T_s is set to 10 seconds by default.
2. If condition 1 is triggered, three short service rate probes are sent, spaced 2 seconds apart, from the player to the SR_{min} hop. Responses are averaged to form

SR_{avg}.

The player process 20 (Figure 1) computes the expected utilization $\rho = 1 - \frac{SR_{avg}}{B}$. If condition 1 is satisfied and the utilization from condition 2 is more than a specified utilization threshold, e.g., 95%, the player process 20 (Figure 1) declares sustained congestion (Confirmed Congestion), and asks the server to enter the constant bit-rate emulation mode. An alternative method may consists of a simple rule where if packet losses exceed a threshold for N_1 consecutive sampling time-windows, the player process 20 (Figure 1) declares sustained congestion and asks the server to enter the constant bit-rate emulation mode. By default, N_1 is set to 1, and time-window is set equal to 1 second.

End of Congestion Rule (EOC Test)

The end of congestion rule is analogous to the congestion detection rule:

1. No packet losses for a consecutive time-window of duration T_e. T_e is set equal to 10 seconds by default.
2. If condition 1 is triggered, three short service rate probes are sent, spaced 2 seconds apart, from the player to the SR_{min} hop. Responses are averaged to form SR_{avg}.

   The player computes the expected utilization $\rho = 1 - \frac{SR_{avg}}{B}$.

If condition 1 is satisfied and the utilization from condition 2 is less than a specified threshold, e.g., 90%, the player declares sustained congestion to be over, and asks the server to transition to the state of recovery scheduling. An alternative method may consists of a simple rule where if in a state of sustained congestion, no packet losses are manifest for N_2 consecutive sampling time-windows, the player declares sustained congestion to be over, and asks the server to transition to the state of recovery scheduling. By default, N_2 may be set equal to five, and the sampling time-window is set equal to 1 second.

Thus, an apparatus and method for predictable and differentiated delivery of multimedia streaming on the Internet is provided. One skilled in the art will appreciate that the present invention can be carried out in other ways and practiced by other than the described embodiments, yet not departing from the spirit and essential characteristics depicted herein. The present embodiments therefore should be considered in all respects as illustrative, and the present invention is limited only by the claims that follow.

CLAIMS

Having thus described our invention, what we claim as new, and desire to secure by Letters Patent is:

1. A method of streaming of multimedia audio and video packets from at least one sending computing device to an at least one receiving computing device via a network, said method comprising the steps of:

a) at a start of said streaming communicating configuration information from a service manager via the network to both an at least one sending computing device for use by an at least one server process and an at least one receiving computing device for use by an at least one player process, said at least one sending computing device comprising an at least one retransmit buffer, an at least one output buffer, an at least one backup buffer and said at least one server process, said at least one receiving computing device, comprising an at least one player buffer and an at least one player process;

b) queuing same packets in the at least one output buffer and to the at least one backup buffer;

c) determining if there are packets in the at least one output buffer and transmitting the packets from the at least one output buffer to the at least one receiving computing device via the network;

d) receiving the packets on the at least one receiving device and storing them in the at least one player buffer;

e) determining if there are omissions in the packets stored in the at least one player buffer and forwarding retransmission requests to the at least one sending computing device and storing them in the at least one retransmit buffer; and

f) determining if there are said retransmit requests in the at least one retransmit buffer and transmitting the packets from the at least one backup buffer to the at least one receiving computing device via the network.

2. The method of claim 1 wherein, the network is Internet.

3. The method of claim 2 wherein, a size of the at least one player buffer is $\geq c*x*e_r*P_{loss}$, where c is a number of sampling intervals in a congestion period, x is a sampling or retransmission interval, $c*x$ is a worst-case duration of congestion, e_r is an encoding bit-rate of a stream of the packets and P_{loss} is a worst case of loss of the packets.

4. The method of claim 3 wherein, a size of the at least one output buffer is $\geq c*x*e_r*P_{loss}$.

5. The method of claim 4 wherein, a size of the at least one backup buffer is $\geq m*x \geq x*\frac{\log\left(0.1*s/e_r*x\right)}{\log P_{loss}}$, where m is a number of retransmits that will reduce the loss rate for older packets to a negligibly small number, and $m*x$ is a minimum required size of the at least one player buffer.

6. A method of detecting and reacting to a network congestion which interferes with streaming of multimedia audio and video packets from at least one sending computing device to an at least one receiving computing device via a network, said method comprising the steps of:

a) communicating configuration information including network parameters from a service manager via the network to the at least one sending computing device and to the at least one receiving computing device, the at least one sending computing device comprising at least one server process, the at least one receiving computing device comprising at least one player process;

b) confirming the network congestion by transmitting and retracting network probes;

Aylo Exhibit 1002
Page 358 of 1405

c) executing at least one congestion control technique to limit degradation in the streaming of the packets;

d) detecting an end of the network congestion by transmitting and retracting said network probes;

e) configuring a rate of streaming of the packets according to said network parameters

7. The method of claim 6, wherein said at least one congestion control technique is the constant bit-rate emulation.

8. The method of claim 7, wherein said at least one congestion control technique only reduces a transmission rate of the packets by the amount of incurred loss of the packets.

9. The method of claim 7, further comprising the steps of:
detecting an end of congestion;
computing available bandwidth in the network; and
transmitting packets according to said bandwidth.

10. The method of claim 9, wherein some of the at least one player process are differentiated and protected from effects of said network congestion to a greater than others by receiving a stream backlog sooner.

11. The method of claim 10, wherein some of the at least one player process are differentiated and protected from effects of said network congestion to a greater than others by receiving a stream backlog sooner according to a rule for transmission rate of the packets

$$e_t = \min(\alpha^j \times SR_{min}, \alpha^j \times \frac{B - \lambda_{other}}{\sum_{i=1}^{i=K} \alpha^i n^i})$$

, where e_t indicates said transmission rate of packets, α^j is a weighting function, for class $\{1 \ldots K\}$.

12. The method of claim 11, wherein said detecting of said end of the network congestion is performed according to the expected utilization $\rho = 1 - \frac{SR_{avg}}{B}$, where SR_{avg} is the average rate at which the packets are transferred from the at least one server process to the at least one player process and B is a bottleneck link capacity at a bottleneck hop.

13. The method of claim 12, wherein said network probes compute an approximate service rate of transferring the packets from said at least one server process to said at least one player process according to the following equation:

$SR^{min}_{m,m-1} = \frac{s}{\frac{s}{B_{m,m-1}} + q^{max}_{m,m-1}}$ and $SR^{avg}_{m,m-1} = \frac{s}{\frac{s}{B_{m,m-1}} + q^{avg}_{m,m-1}}$, where SR_{avg} and SR_{min} are averaged and minimum service rate, $q^{max}_{m,m-1}$ is the maximum queuing delay between a node m and a node m-1.

14. A method of guaranteed delivery of streaming multimedia audio and video packets from at least one sending computing device to an at least one receiving computing device via a network, said method comprising the steps of:

a) continuously processing arriving packets;

b) keeping track of loss of the packets at the at least one receiving computing device, said at least one receiving computing device comprising one or more player processes;

c) determining if said loss exceeds a first threshold during a first interval;

d) verifying congestion symptoms;

e) communicating said verified congestion symptoms to the at least one sending computing device, said at least one sending computing device comprising one or more server processes;

f) monitoring said loss and determining if said loss exceeds a second threshold during a second interval to determine an end of congestion; and

g) verifying congestion symptoms.

15. The method of claim 14, wherein said step of keeping track is performed at user define intervals.

16. The method of claim 15, wherein said first threshold, said second threshold, said first interval, and said second interval are user defined.

17. The method of claim 16, wherein said congestion symptoms are verified by receiving a remote congestion confirmed report.

18. The method of claim 17, wherein said congestion symptoms are verified by performing a congestion probe.

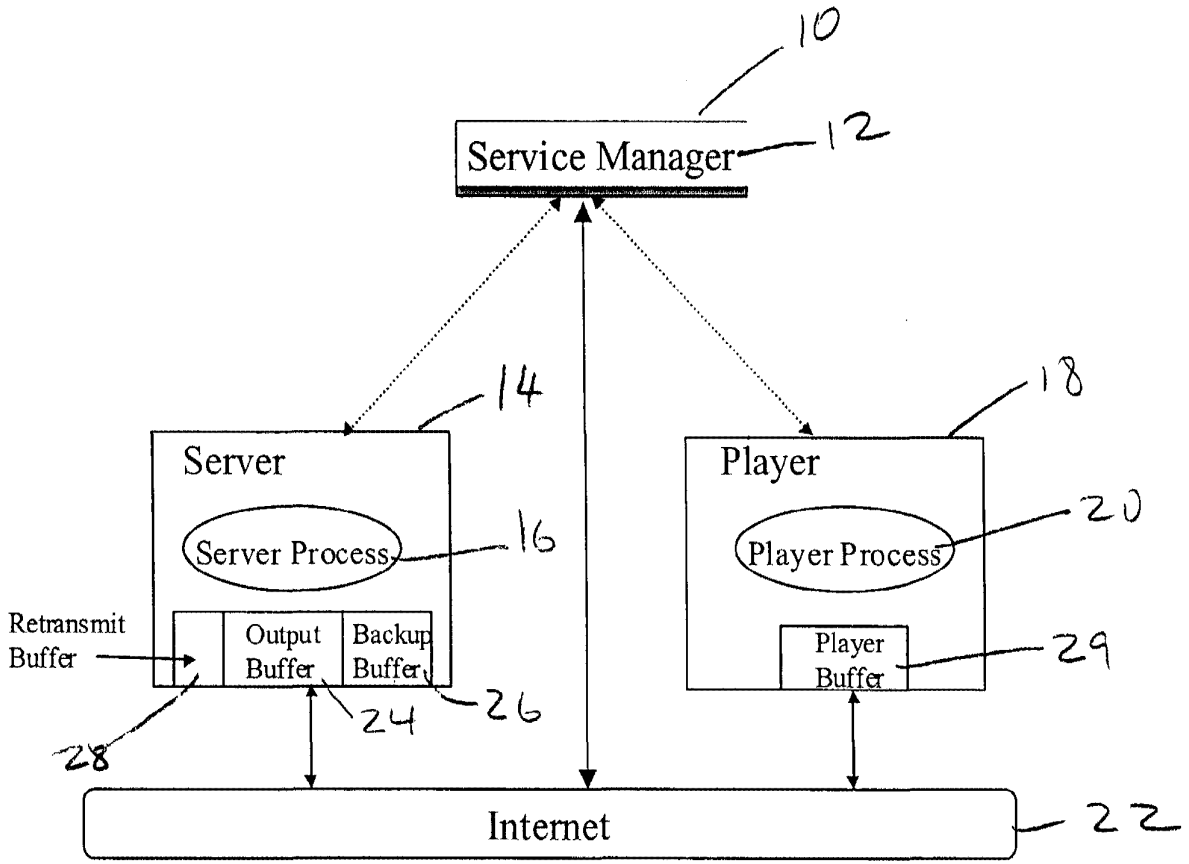

10
Service Manager
12
14
Server
Server Process
16
Retransmit Buffer
Output Buffer
Backup Buffer
24
26
28
18
Player
Player Process
20
Player Buffer
29
Internet
22

Figure 1

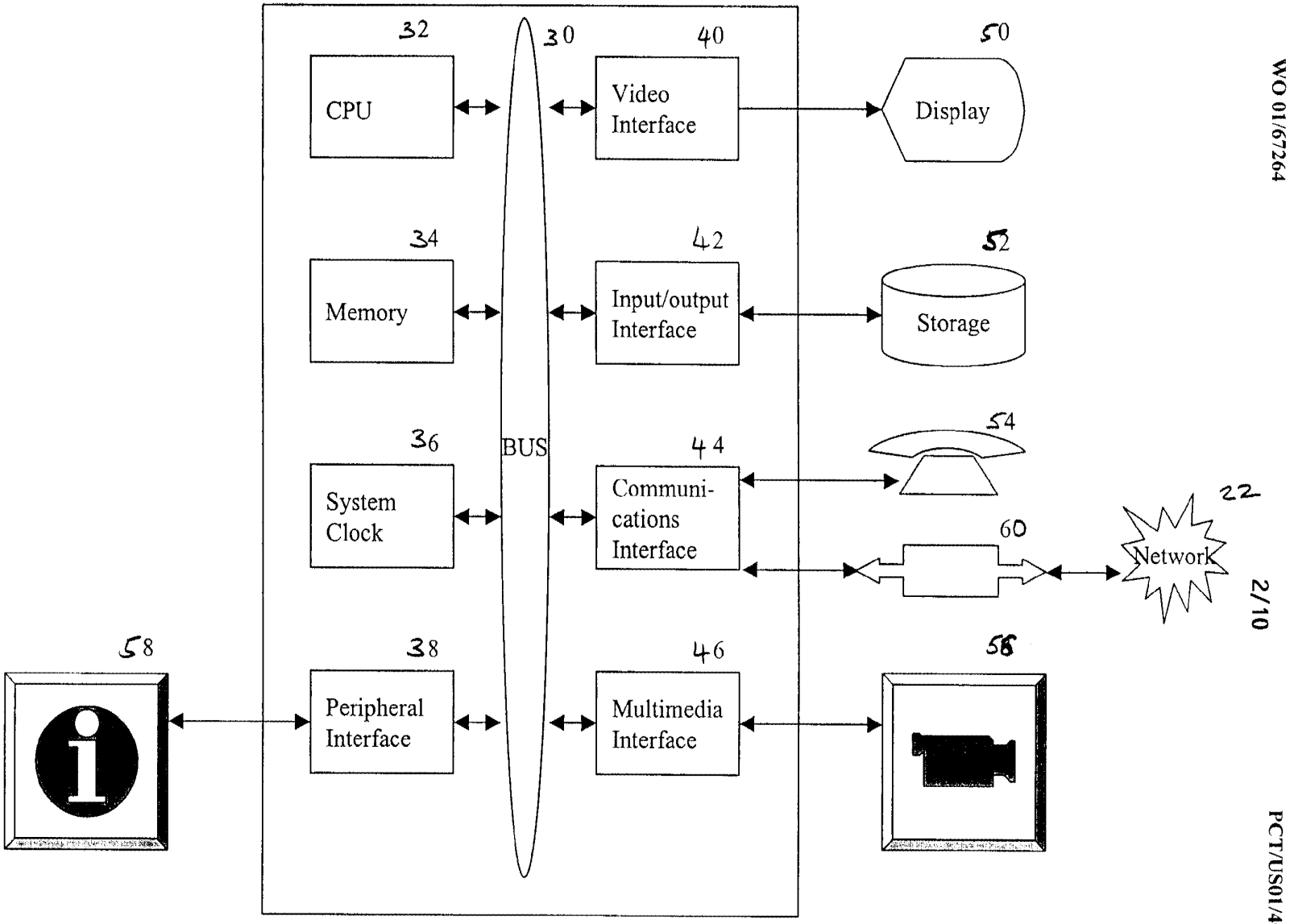

32
30
40
50
CPU
Video Interface
Display
34
42
52
Memory
Input/output Interface
Storage
54
36
BUS
44
22
System Clock
Communi-cations Interface
60
Network
58
38
46
56
Peripheral Interface
Multimedia Interface

Figure 2

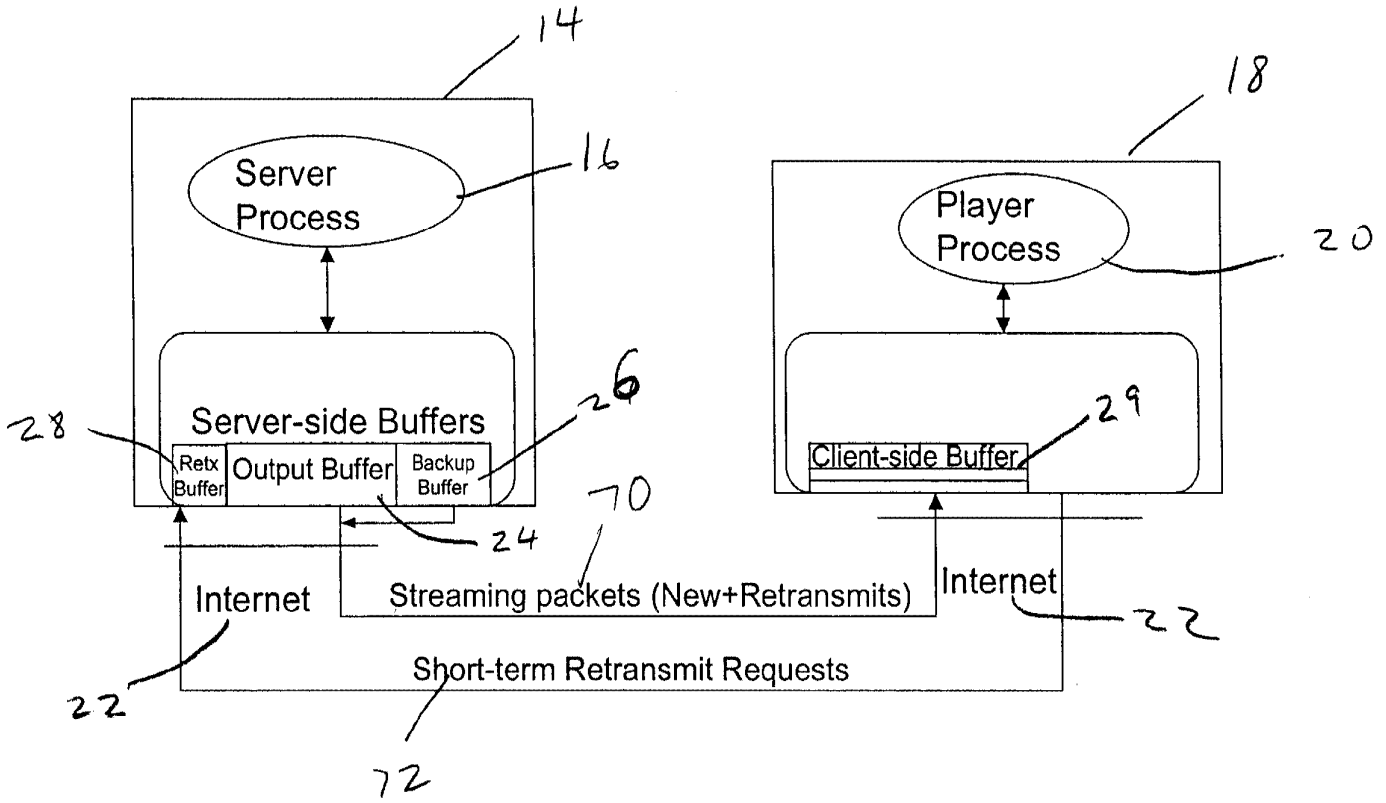

14
Server Process
16
18
Player Process
20
Server-side Buffers
26
28
Retx Buffer
Output Buffer
Backup Buffer
24
Client-side Buffer
29
70
Internet
Streaming packets (New+Retransmits)
Internet
22
22
Short-term Retransmit Requests
72

Figure 3

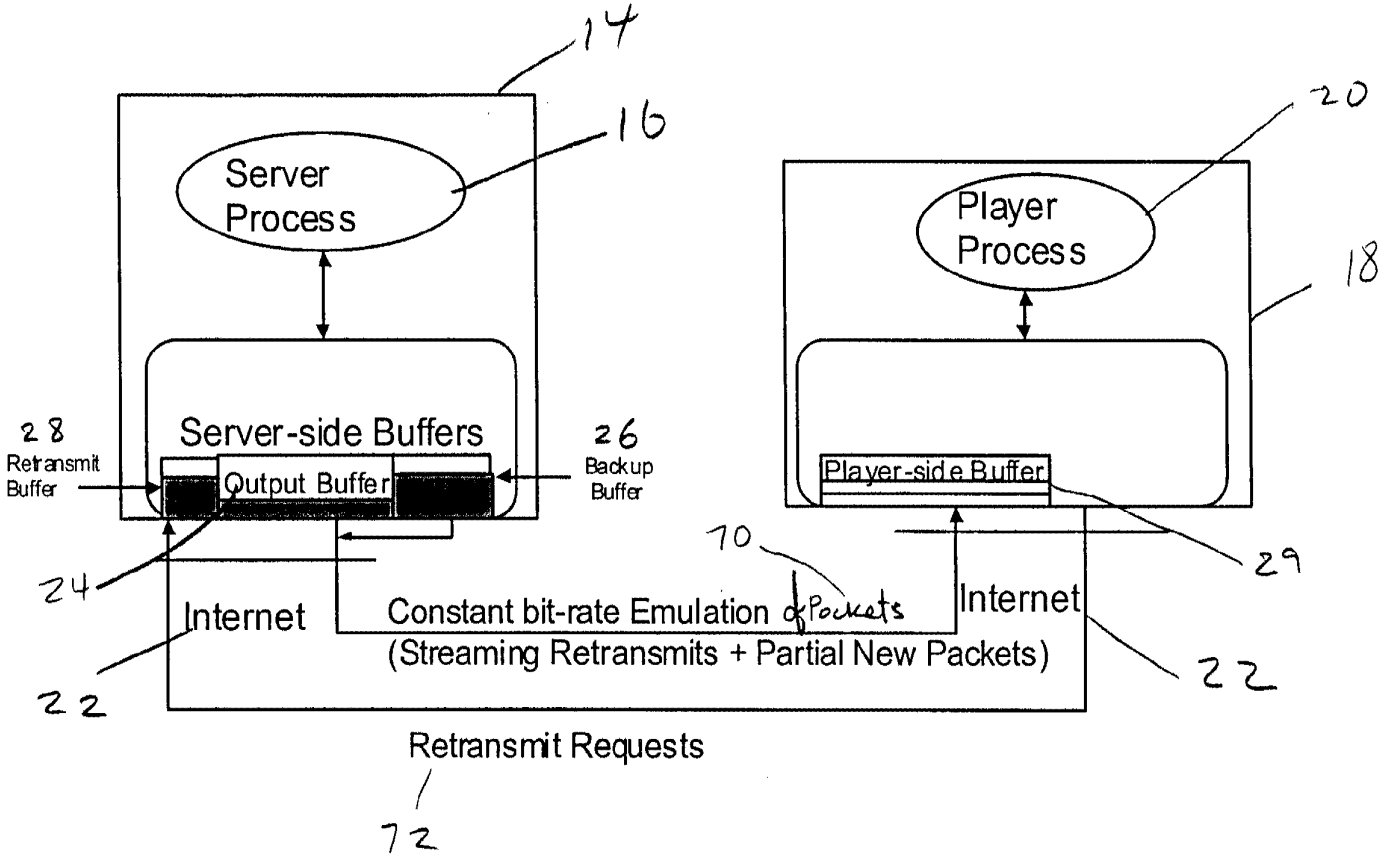

14
16
Server Process
20
Player Process
18
28
Retransmit Buffer
Server-side Buffers
Output Buffer
26
Backup Buffer
Player-side Buffer
29
24
70
Internet
Constant bit-rate Emulation of Packets
(Streaming Retransmits + Partial New Packets)
Internet
22
22
Retransmit Requests
72

Figure 4

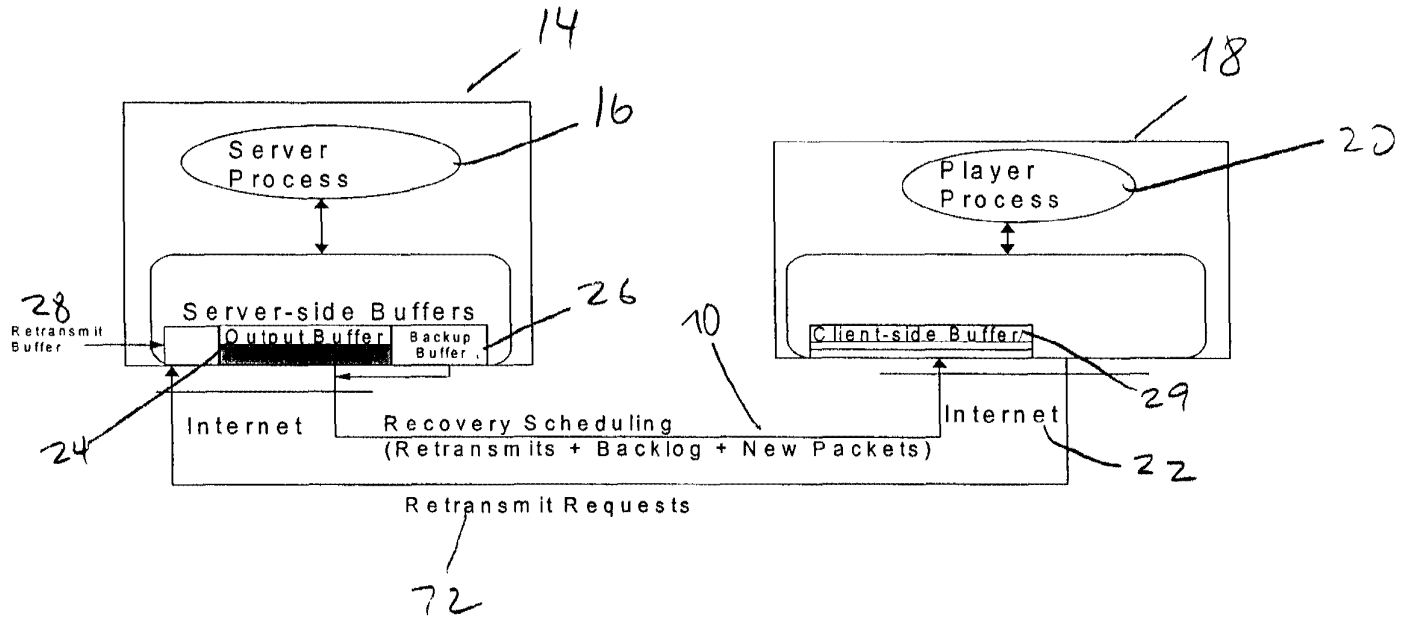

14
16
Server Process
18
20
Player Process
28
Retransmit Buffer
Server-side Buffers
Output Buffer
Backup Buffer
26
10
Client-side Buffer
29
24
Internet
Recovery Scheduling
(Retransmits + Backlog + New Packets)
Internet
22
Retransmit Requests
72

Figure 5

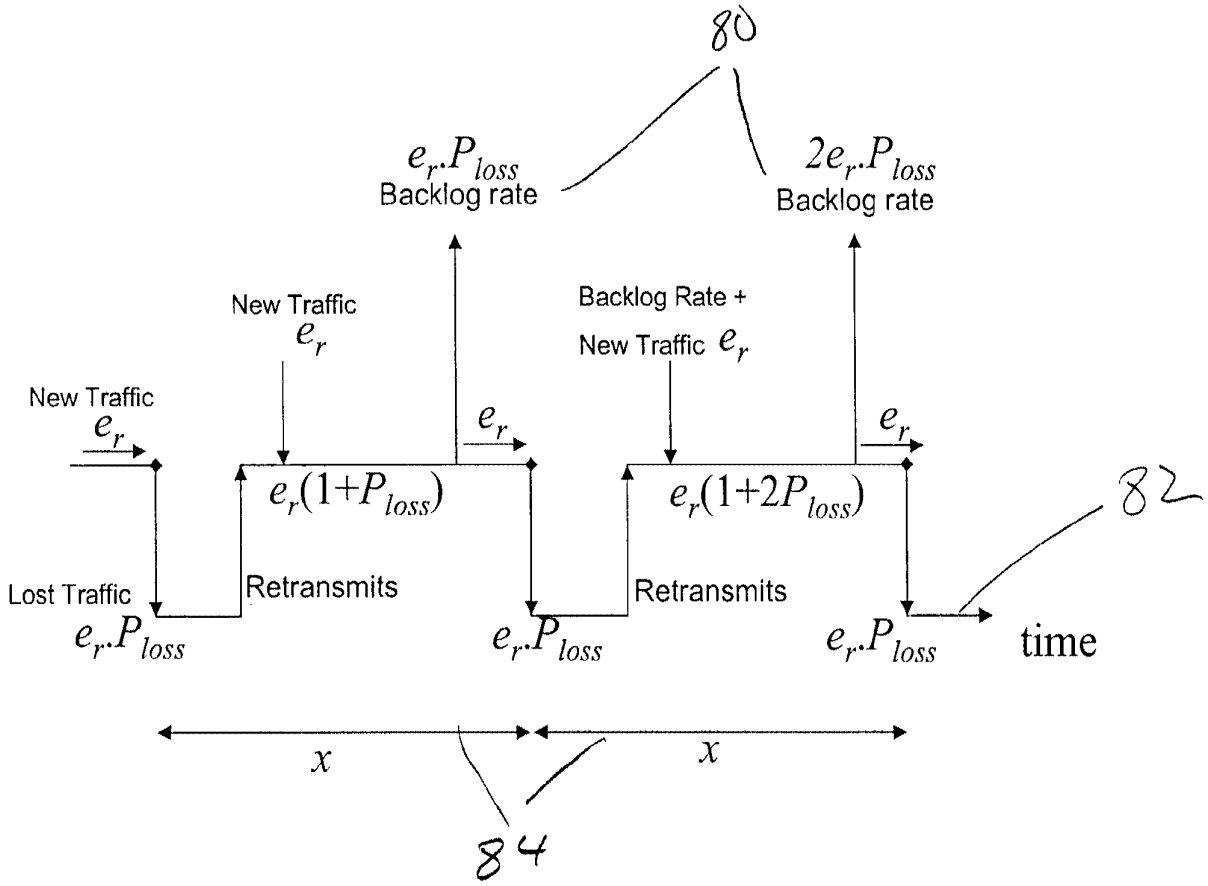

80
$e_r.P_{loss}$
Backlog rate
$2e_r.P_{loss}$
Backlog rate
New Traffic
e_r
Backlog Rate +
New Traffic e_r
New Traffic
e_r
e_r
e_r
$e_r(1+P_{loss})$
$e_r(1+2P_{loss})$
82
Lost Traffic
$e_r.P_{loss}$
Retransmits
$e_r.P_{loss}$
Retransmits
$e_r.P_{loss}$
time
x
x
84

Figure 6

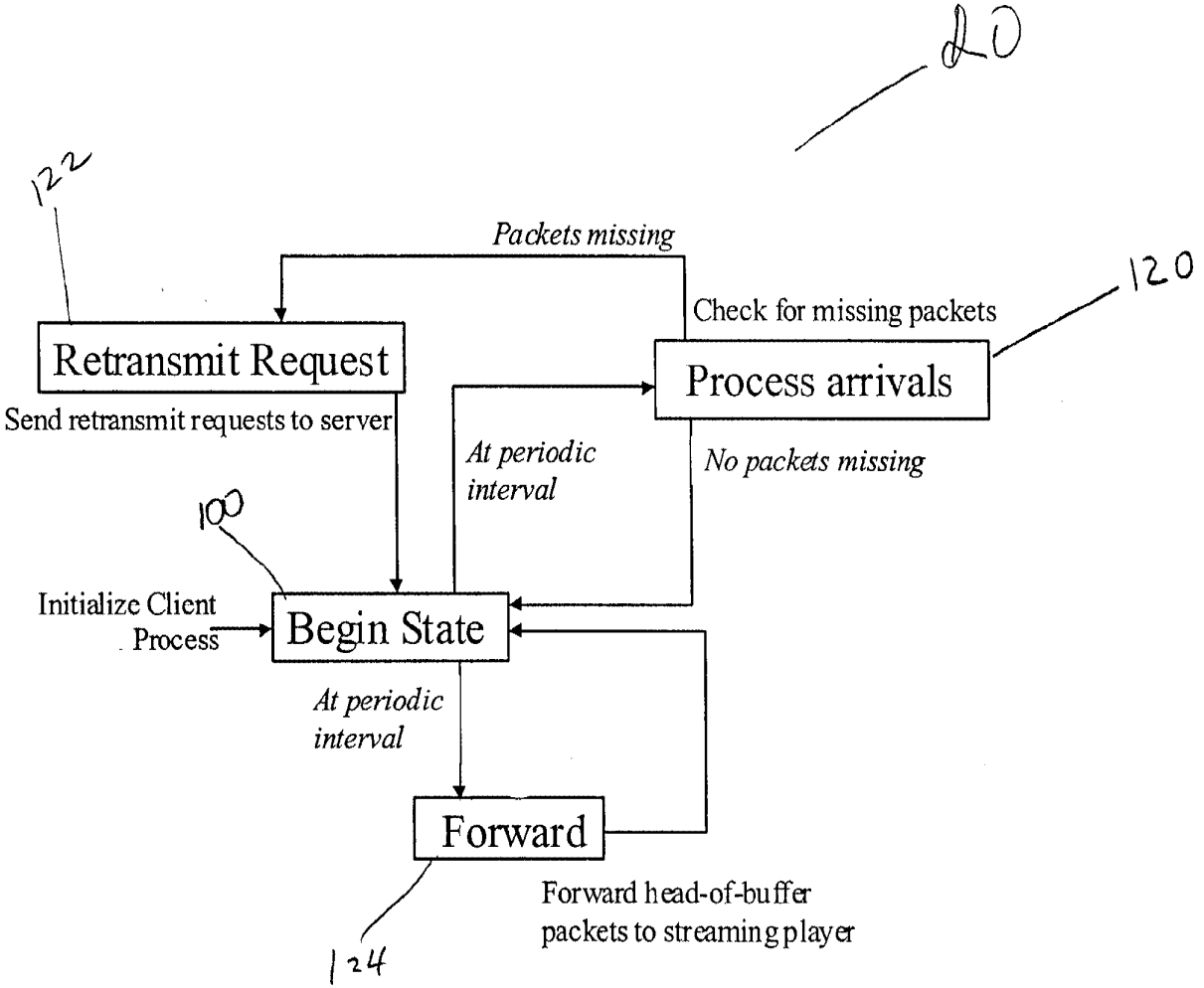

Packets missing
Check for missing packets
Retransmit Request
Process arrivals
Send retransmit requests to server
At periodic interval
No packets missing
Initialize Client Process
Begin State
At periodic interval
Forward
Forward head-of-buffer packets to streaming player

Figure 7

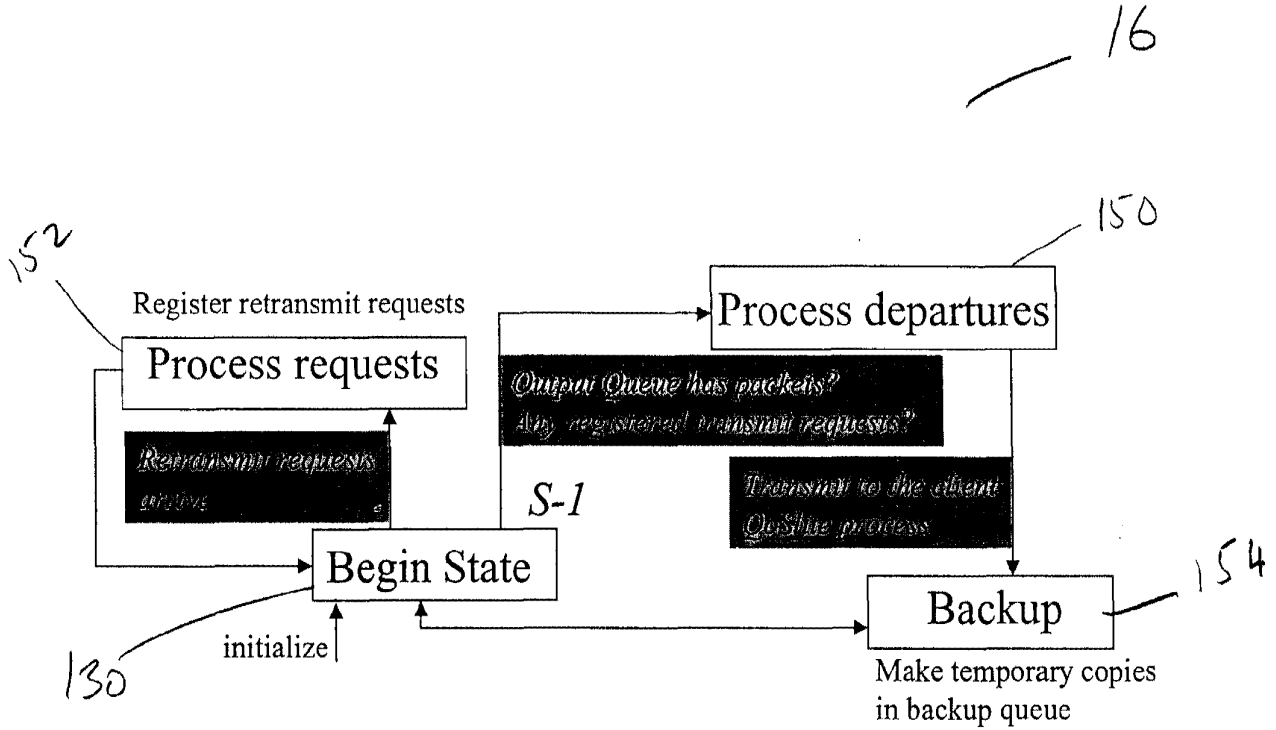

16
152
150
Register retransmit requests
Process requests
Process departures
Output Queue has packets?
Any registered retransmit requests?
Retransmit requests arrive
Transmit to the client QoSlice process
S-1
Begin State
154
Backup
130
initialize
Make temporary copies in backup queue

Figure 8

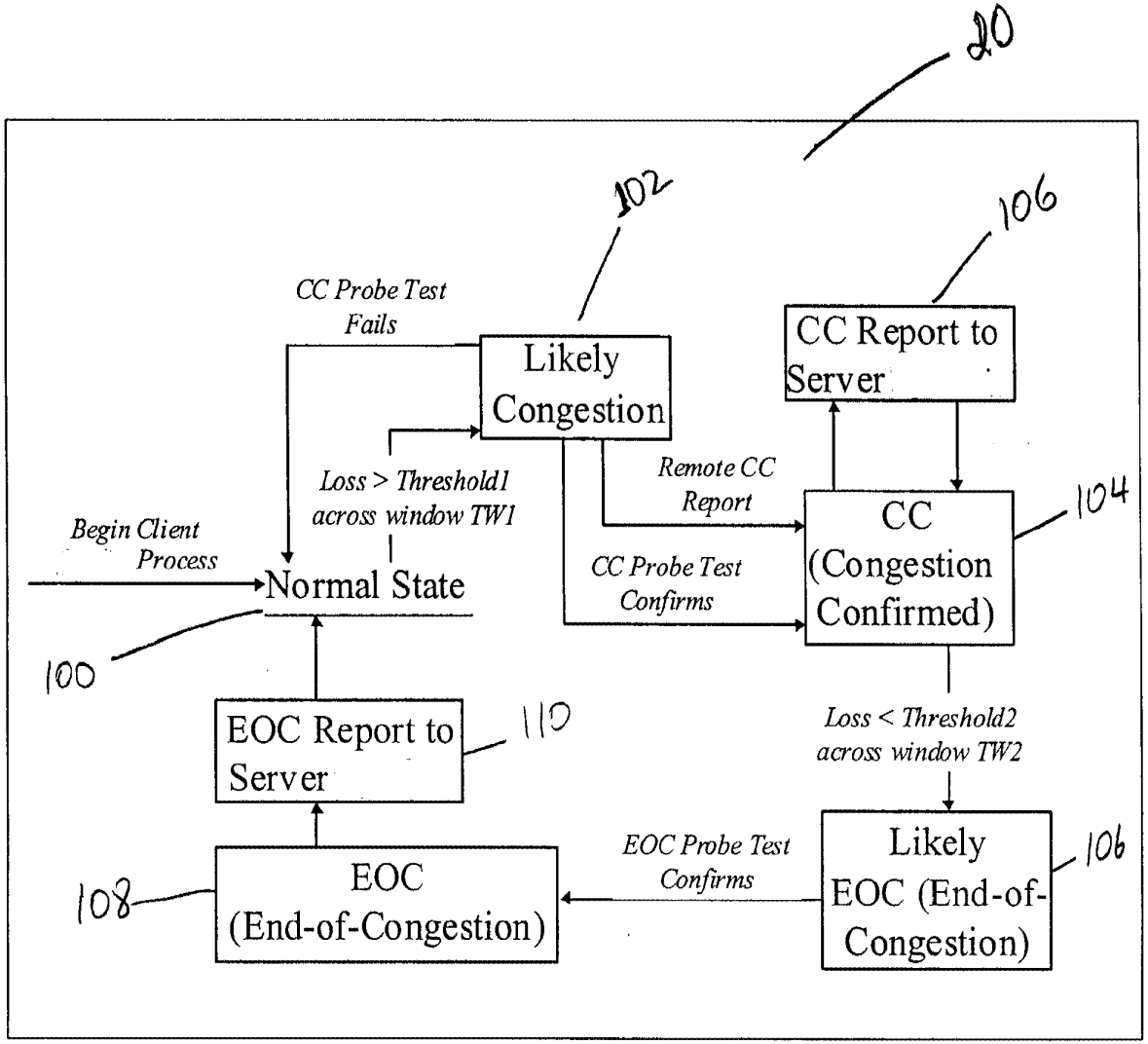

20
102
106
CC Probe Test Fails
Likely Congestion
CC Report to Server
Loss > Threshold1 across window TW1
Remote CC Report
Begin Client Process
Normal State
CC (Congestion Confirmed)
104
CC Probe Test Confirms
100
EOC Report to Server
110
Loss < Threshold2 across window TW2
EOC Probe Test Confirms
Likely EOC (End-of-Congestion)
106
108
EOC (End-of-Congestion)

Figure 9

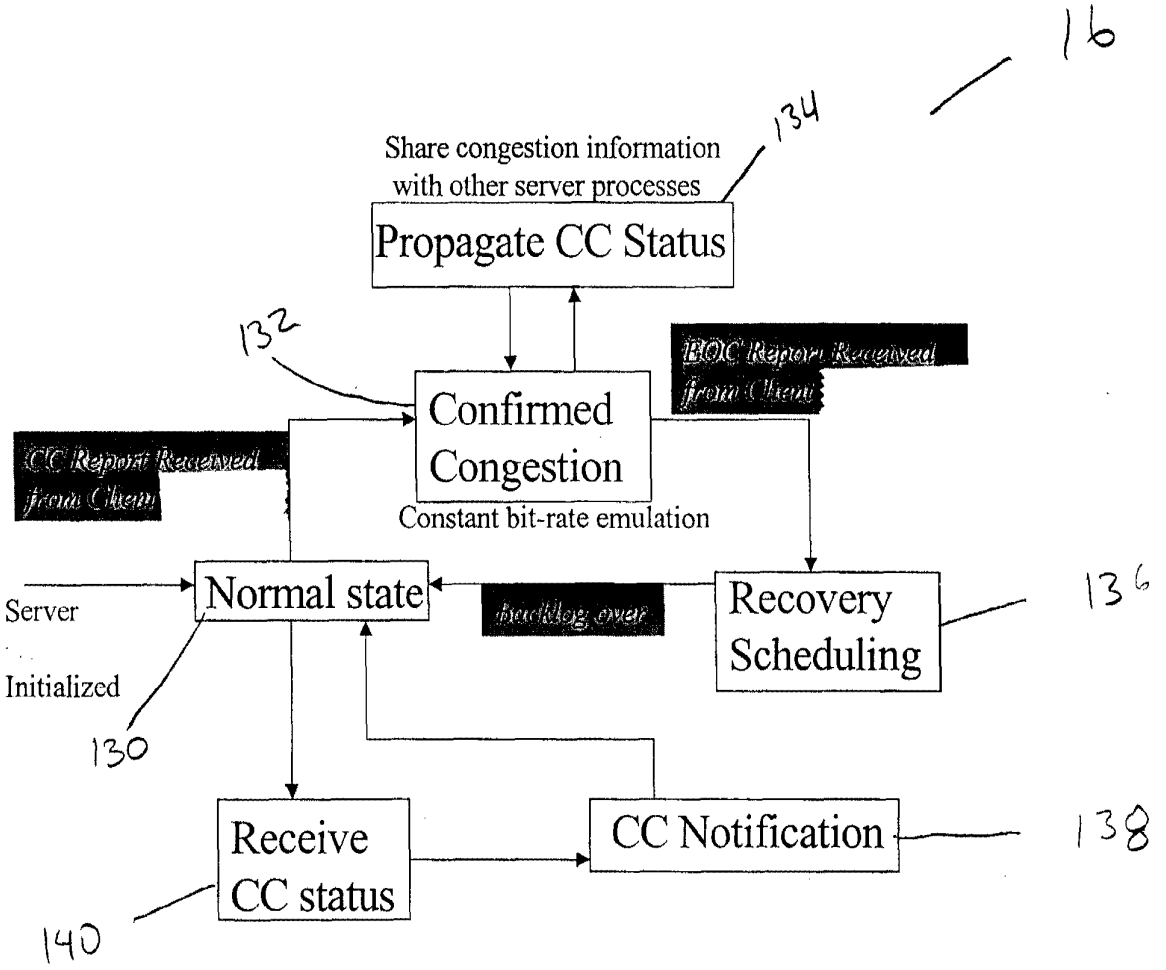

16
Share congestion information
with other server processes
134
Propagate CC Status
132
Confirmed
Congestion
Constant bit-rate emulation
CC Report Received
from Client
BOC Report Received
from Client
Server
Initialized
130
Normal state
backlog over
Recovery
Scheduling
136
Receive
CC status
140
CC Notification
138

Figure 10

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US01/40264

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) :G06F 15/16
US CL :709/231, 232, 235

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. : 709/231, 232, 235

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)

EAST

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X, P	US 6,031,818 A (LO et al) 29 February 2000, col. 2, line 54-col. 5, line 28.	1, 2
X,P	US 6,018,515 A (SORBER) 25 January 2000, col. 4, line 18-col. 10, line 65.	6-10, 14-18
A, P	US 6,014,706 A (CANNON et al) 11 January 2000, see all.	1-18
A	US 5,339,392 A (RISBERG et al) 16 August 1994, see all.	1-18

☐ 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 06 JUNE 2001	Date of mailing of the international search report 27 July 2001 (27.07.01)
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 Peggy Harwood TOD KUPSTAS Telephone No. 703-305-2655

Form PCT/ISA/210 (second sheet) (July 1998) ★

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

(11) EP 1 395 014 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
03.03.2004 Bulletin 2004/10

(51) Int Cl.[7]: **H04L 29/06**, H04L 29/08

(21) Application number: **02019052.6**

(22) Date of filing: **27.08.2002**

(84) Designated Contracting States:
AT BE BG CH CY CZ DE DK EE ES FI FR GB GR IE IT LI LU MC NL PT SE SK TR
Designated Extension States:
AL LT LV MK RO SI

(71) Applicant: **MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.**
Kadoma-city, Osaka 571-8501 (JP)

(72) Inventors:
- **Rey, José Luis**
  64287 Darmstadt (DE)
- **Burmeister, Carsten**
  22081 Hamburg (DE)
- **Hakenberg, Rolf**
  64295 Darmstadt (DE)

(74) Representative: **Grünecker, Kinkeldey,**
Stockmair & Schwanhäusser Anwaltssozietät
Maximilianstrasse 58
80538 München (DE)

(54) A method of transmitting data streams with data segments of variable length

(57) A method of transmitting data streams between a sending host and a receiving host using a transport protocol comprising the steps of making available data streams at the sending host in the form of mono-rate or multi-rate content streams having the same content but different encoding rates, establishing a connection between the sending host and the receiving host, transmitting data streams with a selected encoded rate to the receiving host, and separating the data streams into data segments of variable length at the sending host as required during transmission.

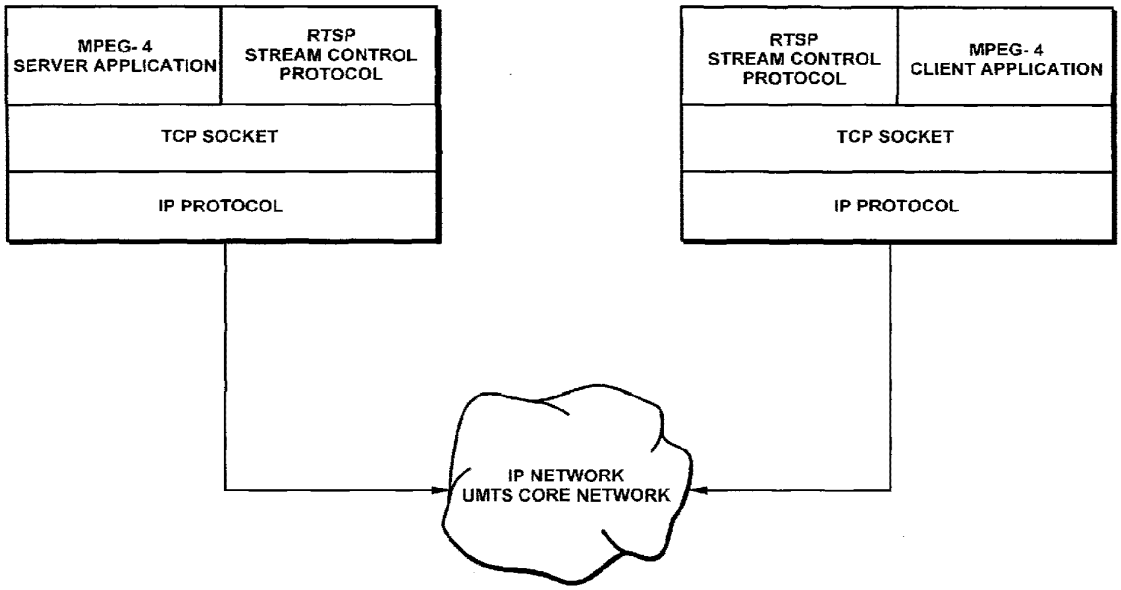


Fig. 1

Printed by Jouve, 75001 PARIS (FR)

EP 1 395 014 A1

Description

[0001] The present invention relates to a method of transmitting data streams between a sending host and a receiving host using a transport protocol such as TCP.

[0002] In network communications between a sending host and a receiving host, the transmission of data packets is mostly done on a best-effort basis. In the communication link, the available bandwidth and the current link conditions vary and hence, the end hosts must be provided with mechanisms to probe and adjust themselves to the current link conditions.

[0003] With the development of 3rd generation networks, so-called streaming applications are gaining importance. Streaming applications are understood as a continuous stream of data, e.g. video or audio data which is typically sensitive to time delays. In order that the user will be able to receive multimedia data streams on a portable receiver, for example a mobile phone, several requirements on the transport mechanism will be imposed in order that the delivered data meets with the resulting time constraints. These constraints are difficult to meet in a best-effort environment, which is subject to bandwidth variations and delays.

[0004] To cope with the problem of bandwidth adaption, it is common praxis to make several data streams with different encoding rates but same content available at the sending host. According to the available bandwidth, it is then switched between the data streams in order to adjust the rate to the available bandwidth. In other words, multi-rate content streams at the sending host are transmitted and switching operations between streams are performed depending on the available bandwidth. The current stream is forwarded to the receiving host in one piece and consequently, for every switching operation, the ongoing connection must be torn down and a new one must be set up. As a result, this operation is inefficient and decreases the data throughput.

[0005] Alternatively, it is possible to transmit only one stream of data to the receiving host and to discard data packets based on priority and time constraints. However, this leads to a loss of data and is therefore not an optimal solution.

[0006] The object subject to the present invention is to minimize the latency of the switching operation and to ensure fast recovery from an incorrect switching decision.

[0007] This object is solved according to the present invention with a method as set forth in claim 1. Preferred embodiments of the method are subject to various dependent claims.

[0008] The method of the present invention supposes the existence of multi-rate content data streams at the sending host. With the method of the present invention, it is possible to quickly switch between data streams that are variably segmented at the application layer as required during transmission. Instead of using data streams of fixed segment length, the data segments given to the socket are made smaller before, during and after the switching. For each content several video streams encoded with different rates are stored at the video server. Under normal sending conditions, i.e. the transmission has reached some equilibrium where the bandwidth is enough and the losses are not dominating, the video content is passed to the sending TCP socket in segments of the same size (irregardless of the stream encoding rate, the segment size is a function of the link, not of the encoding rate). Only when the server is notified (or knows by other means) that a stream switch might be needed, then it reduces the size of these segments.

[0009] Further, with the method of the present invention, it is neither necessary to modify the originally encoded data at the sending host or to modify the transport protocol nor to introduce additional control messages, making the method easy to implement.

[0010] Preferably, the switching is performed between data streams of different rate and the variable segmentation of the data streams is performed around the switching instant. Hence, a quick stream switching from the old to the new data stream can be performed. Further, preferably, the segmentation during, before or after the switching instance is made finer, such that the switching operation is more resilient to an incorrect decision and enables a quicker, further switching operation.

[0011] Advantageously, the step of switching between streams involves a feedback message (e.g. an RTSP command) i.e. communication between hosts when a switching operation is needed and when to make the necessary preparations therefor, for example, to start with finer segmentation.

[0012] It is furthermore of advantage if the feedback message includes at least one of the parameters, which comprise the length of data segment, an identifier of the data to be sent or a stream control command which makes the operation smoother and less susceptible to data losses.

[0013] Preferably, the length of the data segments is gradually increased after a switching instant to reduce interface communication, i.e. between application and TCP socket.

[0014] The present invention will be more fully understood from the following detailed description with reference to the accompanying drawings which show:

Figure 1: the basic architecture of a sending host and a receiving host connected by a communications network in which the method subject to the present invention can be performed,

Figure 2: a block diagram of a server architecture as the sending host in which the method of the present invention is performed, and

Aylo Exhibit 1002
Page 374 of 1405

Figure 3: a block diagram of a client architecture.

[0015] Figure 1 illustrates the basic architecture of a communications network for connecting a video server as a sending host to a video client acting as a receiving host. The communications network is, for example, a UMTS core network, an external network, such as the internet or an internal intranet over which the server and the client communicate by means of a transport protocol. As transport protocol TCP is selected as an example, but it is clear to a skilled person that any other transport protocol for transmitting data can be employed.

[0016] In a more general way, it will now be explained how TCP handles the exchange of data between the server and the client.

[0017] First of all, the video server needs to transfer its data to the protocol and tell it to send same to the client. To progressively read and send data, the protocol uses a kind of buffer with added functionalities, a so-called TCP socket. Generally, sockets are unequivocally identified by the IP addresses of the correspondents, the protocols and ports which are used. Once a connection between two sockets is established, the communication can be controlled by use of interface calls, which, for example may comprise "create a socket", "bind a socket to a transport address", "accept" or "close a connection", "send", "receive" or "write" data between the sender and the receiver.

[0018] Wide-spread socket implementations are UNIX BSD 4.2 and Windows Winsock 2.0, where equivalent interface calls are implemented.

[0019] Each interface call has parameters, e.g. for the "send" call, parameters of relevance are the socket descriptor, the identifier of the data to be sent and the length of the data stream. Once a data stream is passed in one call, no single bit of this data stream can be modified. The only way to abort transfer of this data stream is to break down the connection.

[0020] Nevertheless, in order to close the connection, the application must explicitly issue a "close" call. Interface calls can be queued as, for example, in FTP. When asked for a file, the FTP server usually issues two interface calls for this transmission, namely a "send" call indicating the socket, the identifier of the data stream and the length thereof and a "close" call identifying the socket. The two interface calls are put into the queue and when the "send" call is completed, the connection is irreversibly broken down because the "close" call was next in the queue.

[0021] Alternatively, for example, in case of TELNET, the connection is established and the socket is set to wait for incoming data, i.e. commands typed on the keyboard.

[0022] Of course, the application can be configured such that the server only issues a send call and waits for the client to close. However, this is uncommon.

[0023] Additionally, it is assumed that the TCP protocol provides a means for asynchronously signaling certain information about events in the socket. Often in the specification, the information will be an error message. In other cases, there will be information relating to the completion of processing a SEND or RECEIVE or another interface call.

[0024] Referring again to Figure 1, the video server provides, e.g. an MPEG 4 compressed data stream, i. e. a given video content, which is made available to the TCP socket in the form of mono-rate or multi-rate streams. Multi-rate streams are defined for the purpose of this and the following description, as one or several data streams with the same video content at different encoding rates. The MPEG data streams are forwarded to the TCP socket. Subsequently, by means of an IP protocol, data is transmitted to the video client. A stream control protocol, e.g. RTSP, should be used to perform such operating as PLAY, PAUSE or STOP the transmission.

[0025] With reference to Figure 2, which shows the basic architecture of a video server 200, at the MPEG server application 210, video content is made available in a buffer 211 at different coding rates. A switch 212, which receives stream control RTSP commands from RTSP block 220, such as "Play", "Pause", "Tear down" or "Change rate", selects one of the data streams which is forwarded to the TCP socket 230 in form of an MPEG file.

[0026] It is clear to a skilled person that other stream control protocols than RTSP could be used at the server and the client. The stream control protocol typically instructs to perform play, forwarding, rewinding, pause or stop functions. Additionally, if multi-rate content is available, switch to a higher or lower encoding rate can be instructed.

[0027] Once a server has received a request for a file containing a chosen content encoded at a desired rate, the server application forwards each of the segments, in which the file was previously segmented, one after another to the TCP socket by means of consecutive interface calls, e.g. consecutive SEND calls. Unlike in current FTP applications, the connection is not torn down after each data transmission process is completed. The data connection remains open, so that there is no need to build up a new connection every time a segment is transmitted. Following this, the TCP socket forms with these segments packet data units (PDUs) for transmission to the client, as it finds most appropriate

[0028] Upon establishing a connection, the TCP socket selects a segment size for the data streams which the server finds most appropriate for the link under consideration of the channel bandwidth and starts transmitting.

[0029] Initially, the size of the segments is made small before, during and after a switching operation in order to avoid a large waiting time before a further switch operation can be performed in case the previous switch decision was incorrect. Subsequently, the size of the data segments can be increased gradually throughout the connection time.

Aylo Exhibit 1002
Page 375 of 1405

[0030] Eventually, when there is a change in the available link bandwidth, the video client shall indicate a possible switch operation by means of, for example, an RTSP message and the server will start doing finer segmentation. Hence, the server switches between data streams of different encoding rates. This is done based on initial information and subsequently controlled by feedback from the client. Alternatively, this could be determined by the server, which would, however, impose processing requirements on the server. Also a combination of control mechanisms at the server and the client is possible. How the server or the client exactly makes the decision to perform a switching operation or indicates a possible anticipated switch is out of the scope of the present invention.

[0031] The basic architecture of a TCP streaming client 300 is illustrated in Figure 3. The client 330 typically consists of a TCP socket 310, an application layer buffer 320 and a buffer manager 330 for the application layer buffer.

[0032] The client 300 processes the data segments of variable length as they come. In response thereto, it sends ACKs and performs flow control in order to prevent an overflow of the TCP socket 310 while extracting the maximum bit rate possible from the link. To do this, the TCP socket indicates to the server, with each ACK, how much data it can receive until the socket is full using a window field in the header of the TCP protocol. If needed, the TCP client signals a zero-window telling the server to enter a "Persist" mode wherein no data is sent until the client allows to do so while the connection is maintained. The Persist mode and the signaling of a window field are standard functionalities of the TCP protocol. When the TCP circuit has capacity for more data, it signals this to the server. Streaming control commands are signalled to the server by means of RTSP commands.

[0033] The application layer buffer 320, also called "play-out buffer" receives data from the socket and transmits same to the MPEG client application 340. The manager 330 for the application layer buffer 320 receives a frame request from the MPEG client application and issues a socket request upon considering the state of the application layer buffer. The buffer manager 320 therefore insures that the bit rate with which the TCP socket 310 can deliver the new data to the application layer buffer 320 matches the bit rate at which the application layer buffer is emptied to the MPEG client application 340. The exact control mechanism of the buffer manager is out of the scope of the present invention.

[0034] Finally, from the MPEG client application, the data is transferred to a display 350 for visualization for the user.

Claims

1. A method of transmitting data streams between a sending host and a receiving host using a transport protocol comprising the steps of:

   making available data streams at the sending host in form of mono-rate or multi-rate content streams having same content but different encoding rates,

   establishing a connection between the sending host and the receiving host for the duration of transmission,

   transmitting data streams of the same content with a selected encoding rate to the receiving host, and

   separating the data streams in data segments of variable length at the sending host as required during the established connection.

2. The method according to claim 1, further comprising the steps of switching between data streams of different rate and performing the variable segmentation of the data streams around the switching instant.

3. The method according to claim 1 or 2, wherein the data streams are finer segmented during, before or after the switching instant.

4. The method according to claim 2, wherein the step of switching between streams involves a feedback message.

5. The method according to claim 4, wherein the feedback message includes signalling at least one of the parameters which comprise data segment length, an identifier of the data to be sent or a stream control command.

6. The method according to one of claims 1-5, wherein the receiving host communicates to the sending host when a stream switching is needed.

7. The method according to one of claims 1-6, wherein the sending host determines the timing of the switching instant dependent on the available bandwidth of the communication link.

8. The method according to one of claims 1-7, wherein the length of the data segments is gradually increased after a switching instant.

9. The method according to one of claims 1-8, wherein the received data streams at the receiving host are temporarily stored in an internal buffer.

10. The method according to one of claims 1-9, wherein the receiving host issues stream control commands

Aylo Exhibit 1002
Page 376 of 1405

to the sending host depending on the state of the internal buffer.

Aylo Exhibit 1002
Page 377 of 1405

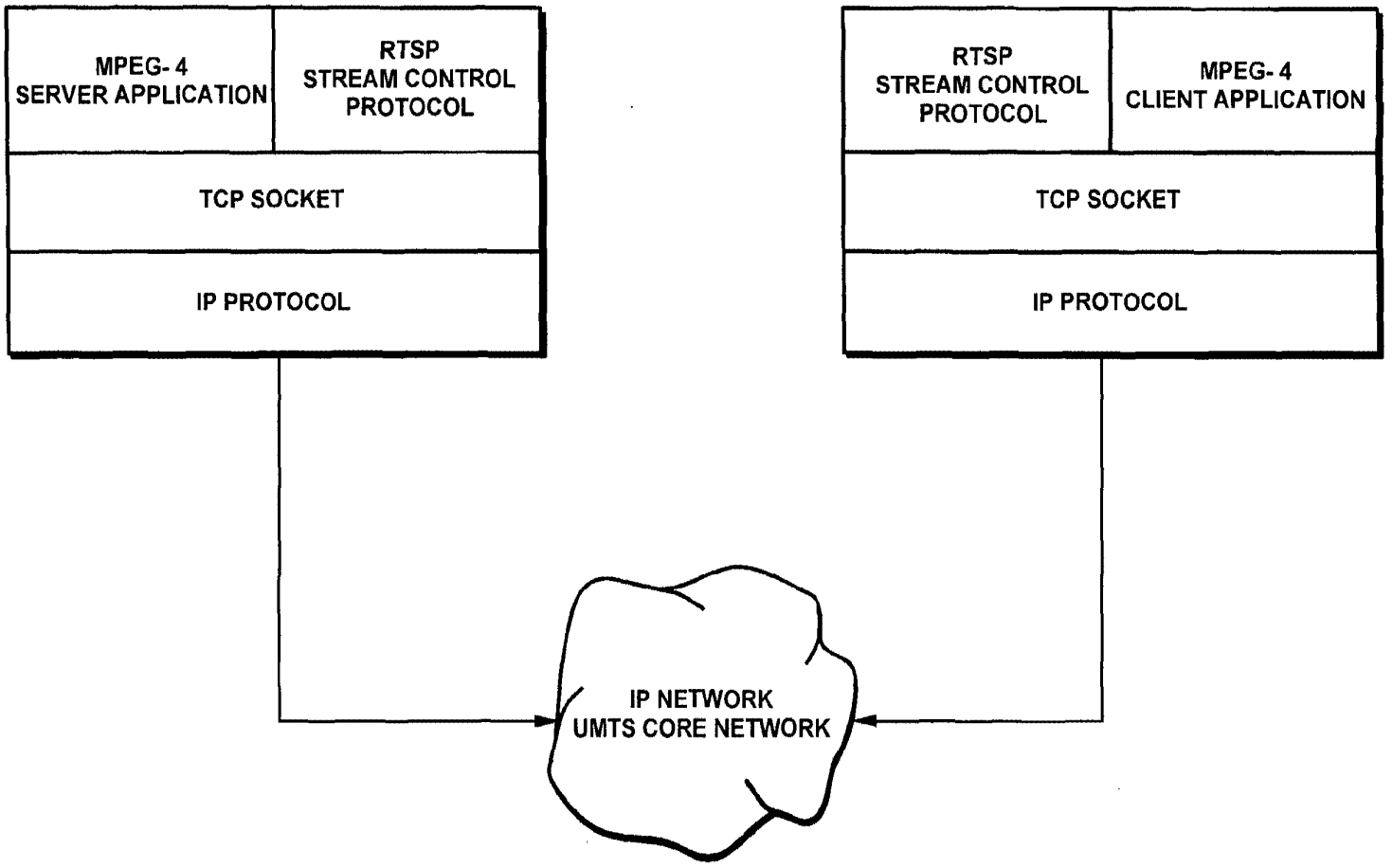

MPEG-4
SERVER APPLICATION
RTSP
STREAM CONTROL
PROTOCOL
TCP SOCKET
IP PROTOCOL
RTSP
STREAM CONTROL
PROTOCOL
MPEG-4
CLIENT APPLICATION
TCP SOCKET
IP PROTOCOL
IP NETWORK
UMTS CORE NETWORK

Fig. 1

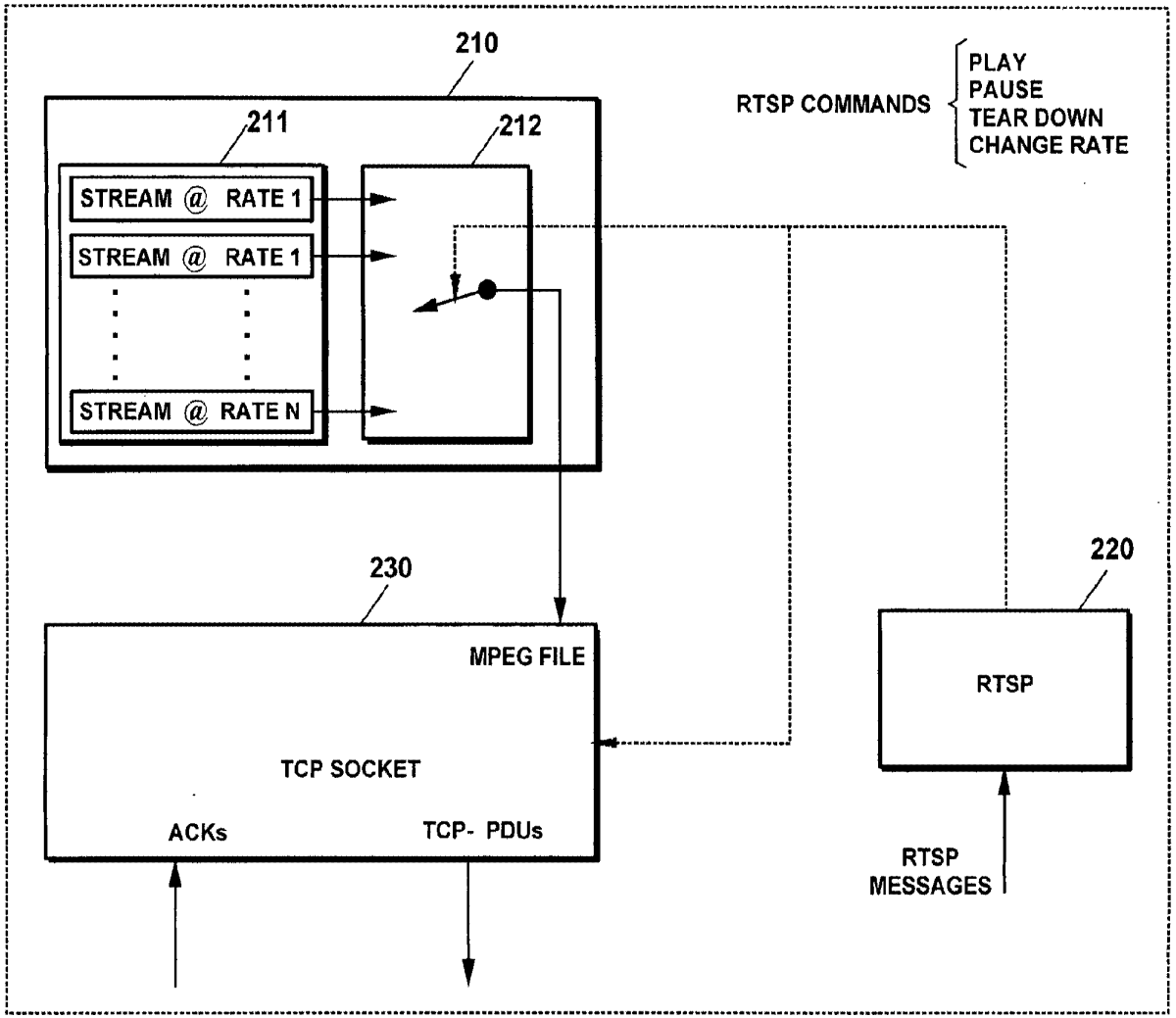


Fig. 2

Aylo Exhibit 1002
Page 379 of 1405

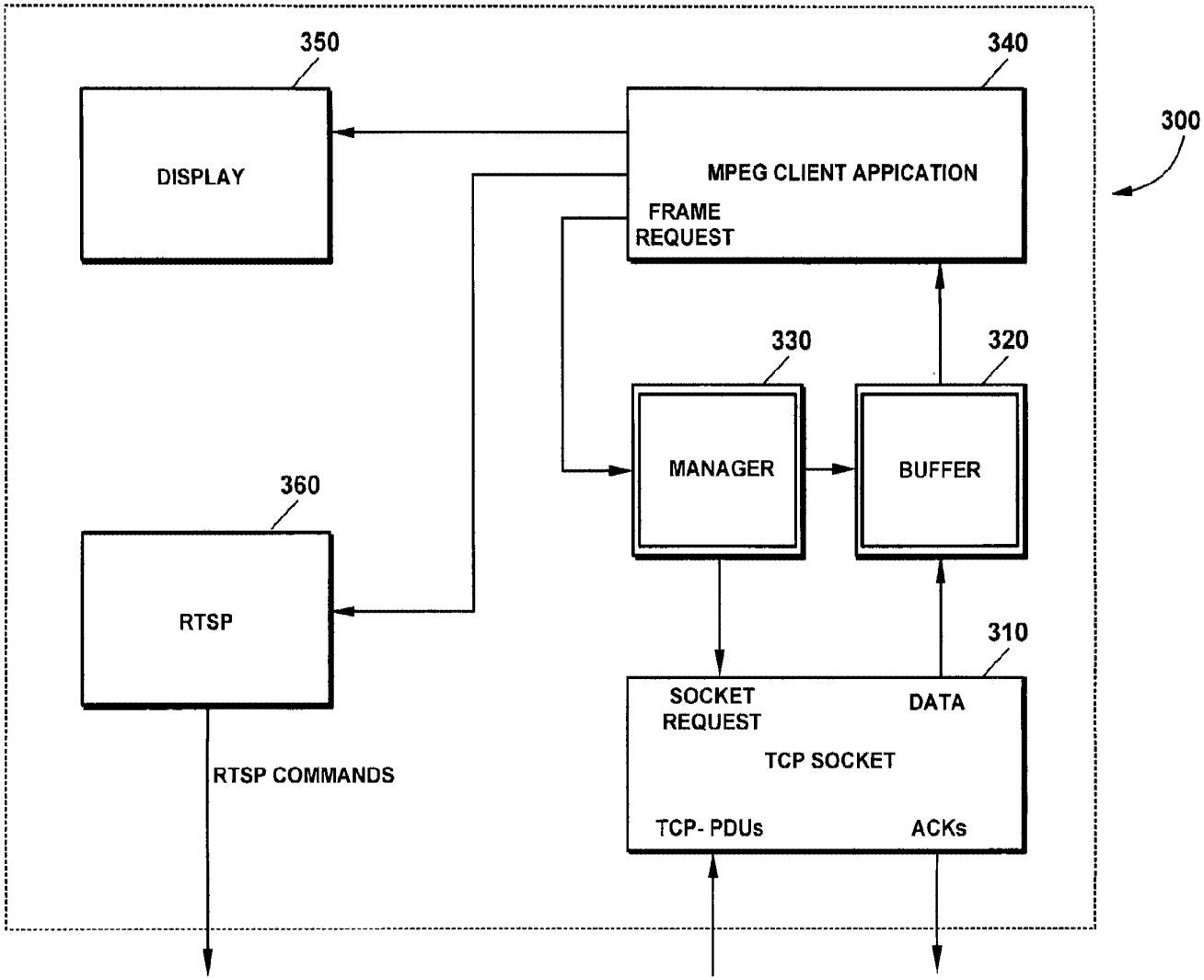

350
340
300
DISPLAY
MPEG CLIENT APPICATION
FRAME
REQUEST
330
320
MANAGER
BUFFER
360
RTSP
310
SOCKET
REQUEST
DATA
TCP SOCKET
RTSP COMMANDS
TCP- PDUs
ACKs

Fig. 3

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 01 9052

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.7)
Y	CONKLIN G J ET AL: "VIDEO CODING FOR STREAMING MEDIA DELIVERY ON THE INTERNET" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE INC. NEW YORK, US, vol. 11, no. 3, March 2001 (2001-03), pages 269-281, XP001093477 ISSN: 1051-8215 * page 271, left-hand column, line 35 - line 44; figures 1,2 * * page 270, left-hand column, line 11 - line 20 * ---	1-10	H04L29/06 H04L29/08
Y	EP 0 786 885 A (NOKIA TELECOMMUNICATIONS OY) 30 July 1997 (1997-07-30) * abstract * * page 2, column 2, line 14 - line 34 * ---	1-10	
X	WO 02 054284 A (LEISTAD GEIRR I ;DANKWORTH HARALD (NO); FAST SEARCH & TRANSFER ASA) 11 July 2002 (2002-07-11) * page 2, line 31 - page 3, line 7 * * page 4, line 6 - page 5, line 6; figure 1 * * page 9, line 18 - page 10, line 15 * * page 11, line 3 - line 28; figure 3 * --- -/--	1	

TECHNICAL FIELDS SEARCHED (Int.Cl.7)
H04L

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	7 January 2003	Michael, T

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)

Aylo Exhibit 1002
Page 381 of 1405

European Patent Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 01 9052

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.7)
A	HUI J Y ET AL: "CLIENT-SERVER SYNCHRONIZATION AND BUFFERING FOR VARIABLE RATE MULTIMEDIA RETRIEVALS" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE INC. NEW YORK, US, vol. 14, no. 1, 1996, pages 226-237, XP000548824 ISSN: 0733-8716 * page 228, left-hand column, line 15 - line 33 * * page 228, right-hand column, line 14 - line 32 * ---	10	
A	JONGWON KIM ET AL: "TCP-friendly Internet video streaming employing variable frame rate encoding and interpolation" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, vol. 10, no. 7, October 2000 (2000-10), pages 1164-1177, XP002226167 ISSN: 1051-8215 * page 1164, right-hand column, line 25 - line 40; figure 1 * ---	7	TECHNICAL FIELDS SEARCHED (Int.Cl.7)
A	WO 02 052859 A (DURRANT SIMON ;LAKE JULIAN (GB); MULROY PAT (GB); PA CONSULTING SE) 4 July 2002 (2002-07-04) * page 3, line 12 - line 30 * -----	7	

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
MUNICH	7 January 2003	Michael, T

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)

Aylo Exhibit 1002
Page 382 of 1405

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 02 01 9052

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

07-01-2003

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 0786885 A	30-07-1997	FI 960382 A	27-07-1997
		EP 0786885 A2	30-07-1997
		US 5914960 A	22-06-1999
WO 02054284 A	11-07-2002	NO 20010056 A	05-07-2002
		WO 02054284 A1	11-07-2002
WO 02052859 A	04-07-2002	WO 02052859 A1	04-07-2002

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

Aylo Exhibit 1002
Page 383 of 1405

(12) UK Patent Application (19) GB (11) 2 367 219 (13) A

(43) Date of A Publication 27.03.2002

(21) Application No 0023062.3

(22) Date of Filing 20.09.2000

(71) Applicant(s)
Vintage Global
(Incorporated in the British Virgin Islands)
c/o Ernst & Young Trust Corp (BVI) Ltd, P.O Box 3340,
Road Town, Tortola, British Virgin Islands

(72) Inventor(s)
Benjamin Neeman
Yossi Rubner
Nehemia Davidson
Uri Salomon

(74) Agent and/or Address for Service
Geoffrey L Melnick
c/o Solo Photographic, 12 Catherine Street,
ST ALBANS, Herts, AL3 5BX, United Kingdom

(51) INT CL[7]
H03M 7/30 , H04L 1/00

(52) UK CL (Edition T)
H4P PDCFM

(56) Documents Cited
EP 1049074 A1
WO 98/09387 A1
US 5847760 A
WO 99/00984 A1
WO 00/72601 A2

(58) Field of Search
UK CL (Edition S) H4P PDCFM PDCFP PDCFX PF
INT CL[7] H03M 7/30 , H04L 1/00 , H04N 7/26 7/32
Online: WPI, EPODOC, JAPIO

(54) Abstract Title
Streaming of media file data over a dynamically variable bandwidth channel

(57) A method of transmitting data streams over a variable bandwidth channel, comprising modifying the quantity of data to be injected into the channel to accord with a measured capacity or bandwidth of the channel. An input stream is passed through a plurality of parallel compressors 40.1-40.4 which each produce an output stream of a different rate to the other streams. Each stream preferably comprises key or reference frames and delta frames. The reference frames are synchronized between the streams so that a data unit in one stream contains the same information as the corresponding data unit in another stream, but is compressed to a different extent. A data stream for transmission along the channel is produced by selecting data units from more than one compressor output stream so as to produce a stream which makes optimum use of the available channel bandwidth. In one embodiment the duration of the data units are one of a group comprising fixed predetermined durations and duration determined by changes in the channel bandwidth. The method is useful in sending multimedia data streams such as audio and video over a channel such as a cellular telephone network or the internet.

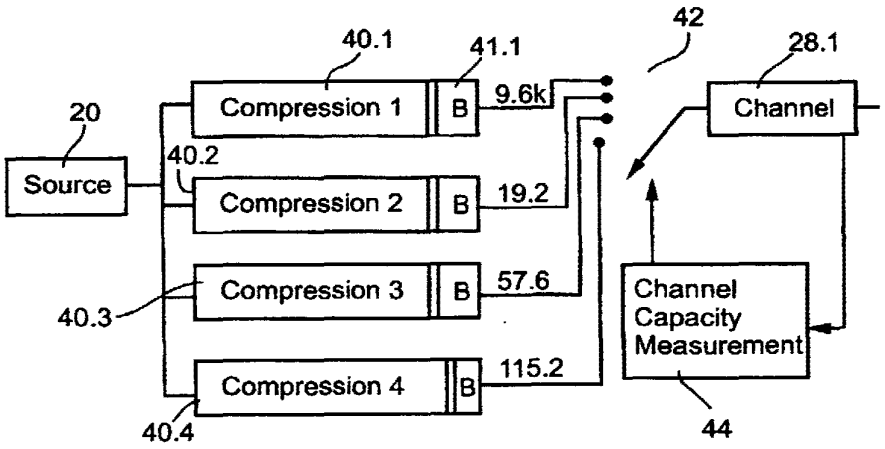


Fig. 4

GB 2 367 219 A

At least one drawing originally filed was informal and the print reproduced here is taken from a later filed formal copy.

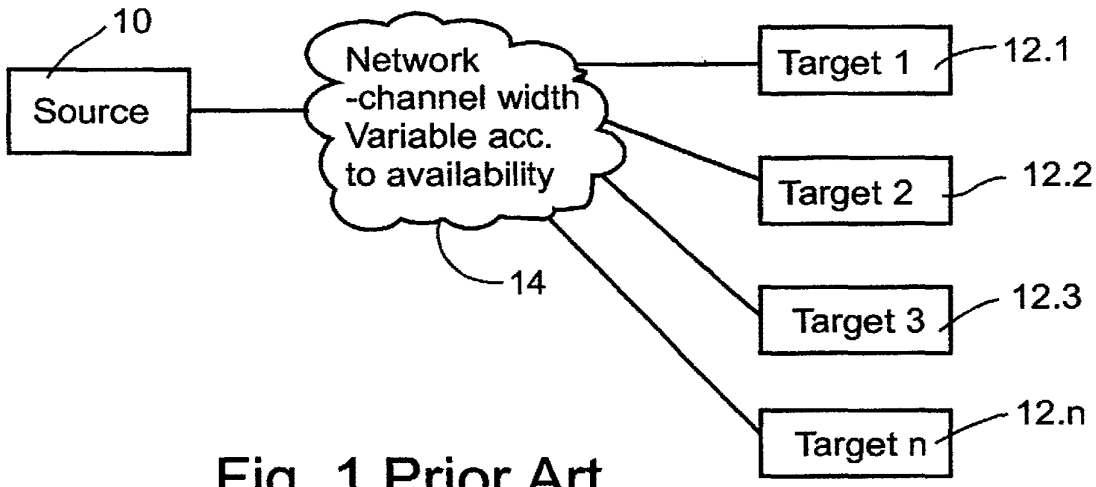


Fig. 1 Prior Art

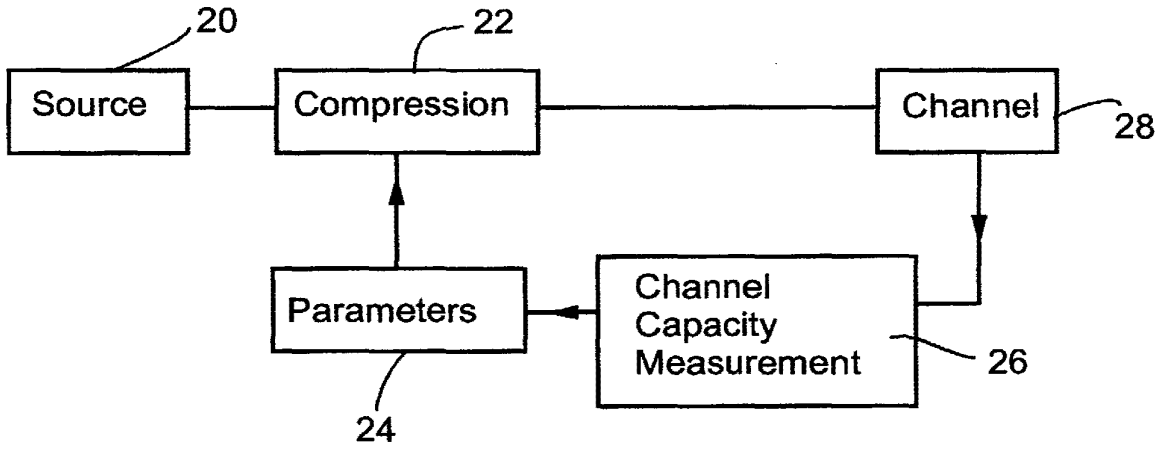


Fig. 2

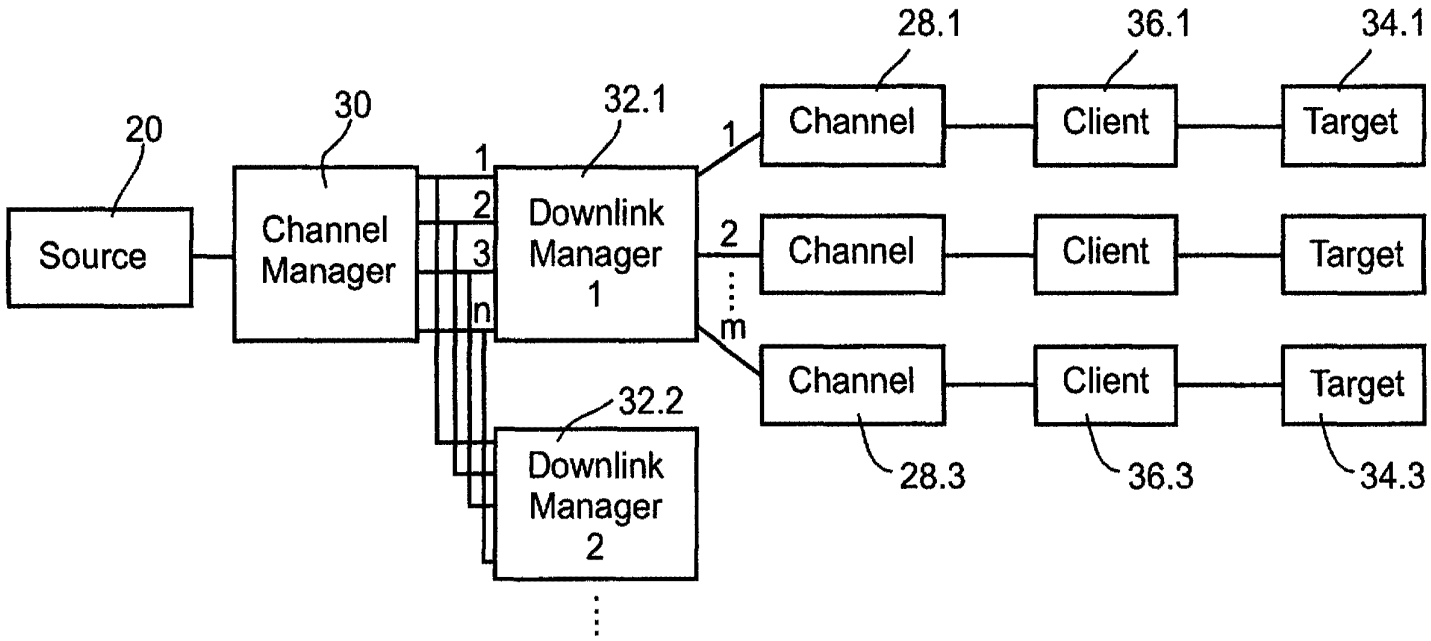

Source
20
Channel Manager
30
Downlink Manager 1
32.1
Downlink Manager 2
32.2
1
2
3
n
1
2
m
Channel
Client
Target
28.1
36.1
34.1
Channel
Client
Target
Channel
Client
Target
28.3
36.3
34.3

Fig. 3

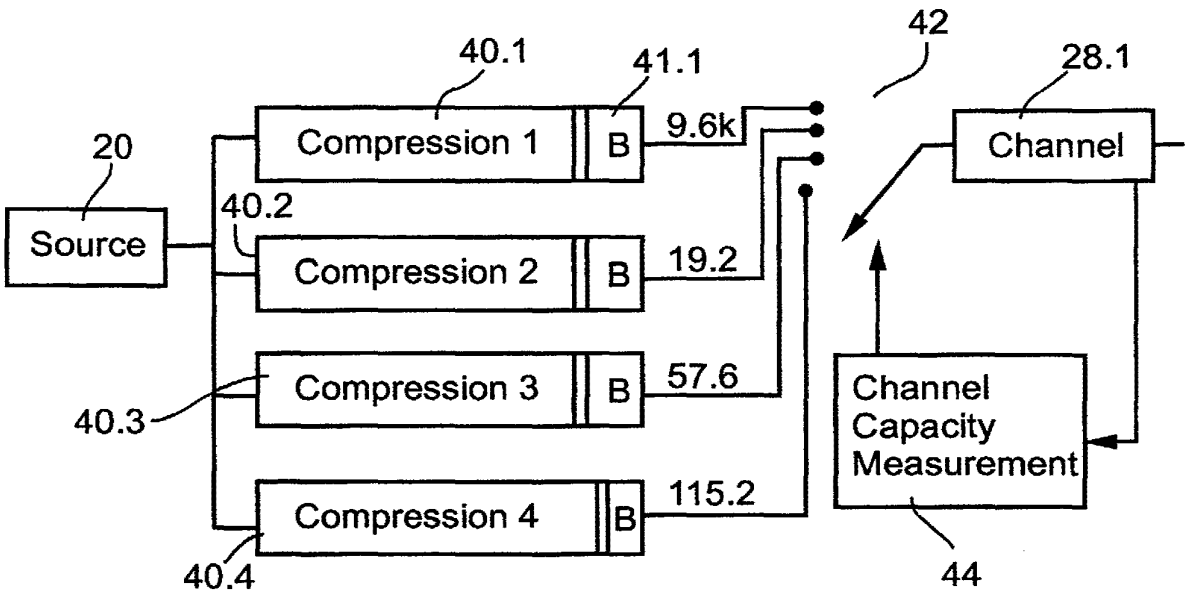

40.1
41.1
42
28.1
20
Compression 1
B
9.6k
Channel
40.2
Source
Compression 2
B
19.2
Compression 3
B
57.6
40.3
Channel
Capacity
Measurement
Compression 4
B
115.2
40.4
44

Fig. 4

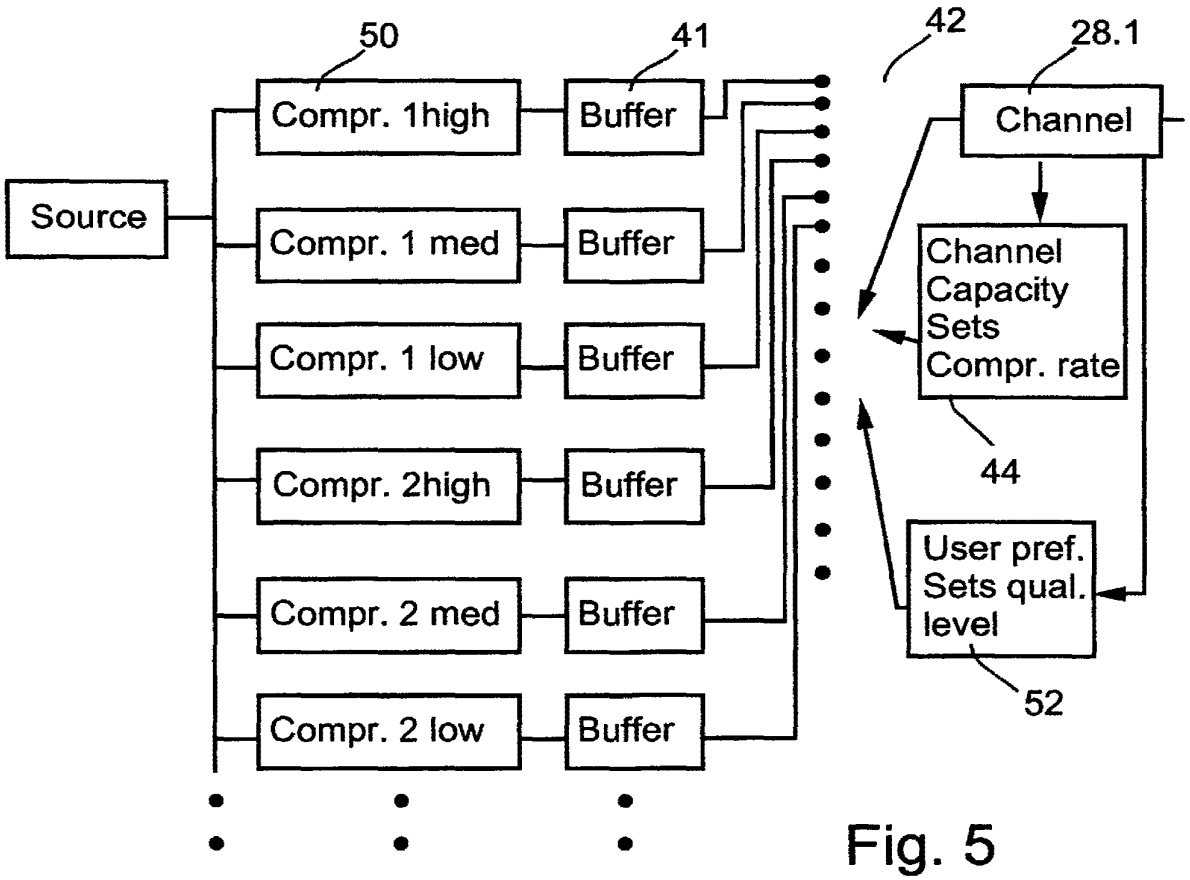

50
41
42
28.1
Compr. 1high
Buffer
Channel
Source
Compr. 1 med
Buffer
Channel
Capacity
Sets
Compr. rate
Compr. 1 low
Buffer
44
Compr. 2high
Buffer
User pref.
Sets qual.
level
Compr. 2 med
Buffer
52
Compr. 2 low
Buffer

Fig. 5

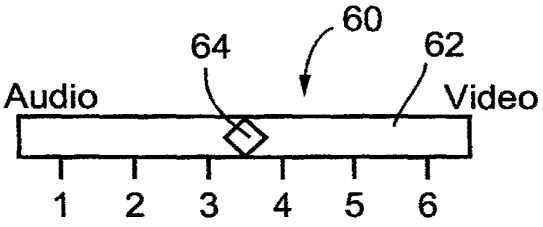


Fig. 6

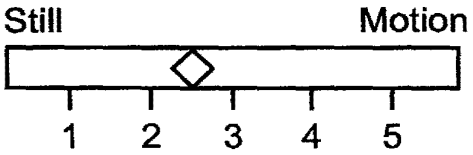


Fig. 7

00/20757 -UK. 9/12/00

Title of the Invention

Streaming of Media File Data Over a Dynamically Variable Bandwidth Channel

Field of the Invention

The present invention relates to streaming of media file data over a channel having a dynamically variable bandwidth, and more particularly but not exclusively to streaming of multimedia data such as video and /or audio data over a channel having a dynamically variable bandwidth.

Background of the Invention

Dynamically varying channel bandwidths are particularly encountered over the Internet and like networks and over radio and cellular telephone links. On the Internet the total bandwidth is fixed and the bandwidth available per channel tends to depend on the amount of usage, the greater the usage the smaller the bandwidth available per user. In radio and cellular links the available bandwidth depends on usage, but the overall capacity may also be dependent on environmental factors.

The Internet relies on packet switched communication, whereas radio and cellular telephone links may be either circuit-switched or packet switched. Either way, the same considerations of channel bandwidth broadly apply.

As the bandwidth changes, a number of approaches are currently in use. One of these is simply to buffer received video frames, to start to show the frames as soon as the buffer is full, and to continue to show the video until the buffer is empty, then to halt the display until the buffer is full again. Another method is to show the video at a real time rate, displaying all frames that arrive in time and dropping all frames that arrive too late.

In both of these methods, data is sent at the same high rate. In the first method, unacceptable delays in showing the data are experienced. In the second method, valuable channel bandwidth is wasted transmitting data that does not get used.

Whether the network is a cellular telephone network or the Internet or like wide area data network, the bandwidth of the data network is variable. In all cases the

larger the number of users the lower the bandwidth per user, and radio-based networks may also suffer from reduced bandwidth due to environmental conditions. Furthermore, in the case of cellular telephone networks, events such as handover between two base stations may also affect the available bandwidth.

In general, cellular networks are able to absorb changes in bandwidth without the user noticing a change in the quality of the voice signal. However, it is desirable to go beyond mere voice communication over cellular telephone networks. Already, low level data capability is being built into mobile telephones and it is envisaged that within the next few years, mobile telephones will be able to receive and display multimedia data. Multimedia data uses large amounts of bandwidth and generally requires to be displayed in real time. Thus bandwidth fluctuations will be immediately noticeable. Currently, however, transmission of video data over cellular phone channels is not being attempted on a commercial basis.

At present, mobile telephone systems are on their second generation and a third generation is on the drawing board but will not be available for some years. The problem is particularly associated with the design of advanced second generation equipment (generation 2.5) which, it is envisaged will be able to provide multimedia within the limitations of second generation capacity. It should be noted however that the same considerations apply to further generations of mobile telephones and also to other environments in which it is desired to transmit multimedia data, such as the Internet.

Generally, multimedia data is sent over a network as a datastream in compressed form. There are numerous algorithms for compressing data, and different forms of compression are appropriate for compressing different forms of data. Furthermore, a single form of compression may give rise to different levels of compression according to compression parameters that may be set. The greater the compression the lower will be the resulting data rate and a high level of compression may be used to overcome channel capacity constraints. A solution may involve selection of the compression parameters to meet a likely worst-case scenario. However, the solution is not complete as a high level of compression leads to low quality output and no advantage is taken of the availability for most of the transmission time of a higher bandwidth availability.

US Patent No. 6,014,694, Aharoni et al, discloses a system for adaptively transporting video over networks wherein the bandwidth varies with time. The

Aylo Exhibit 1002
Page 390 of 1405

system comprises a video/audio codec that functions to compress, encode, decode and decompress video streams that are transmitted over networks having available bandwidths that vary with time and location. Depending on the channel bandwidth, the system adjusts the compression ration to accommodate a plurality of bandwidths ranging from 20kbps for POTS to several Mbps for switched LAN and ATM environments. Bandwidth adjustability is provided by offering a trade-off between video resolution, frame rate and individual frame quality. The system generates a video data stream comprising of key, P and B frames from a raw video source. Each frame type is further comprised of multiple levels of data representing varying degrees of quality. In addition, several video server platforms can be utilized in tandem to transmit video/audio information with each video server platform transmitting information for a single compression/resolution level.

P and B frames are delta frames that depend on previous frames to allow a complete picture to be generated. Thus it is not possible simply to mix groups of frames of different quality levels since the deltas between frames are different. The above US Patent solves the problem by taking advantage of what is known as a group of pictures (GOP) which is a self-contained unit such as a scene. Because they are self-contained, GOPs of different compression levels can be successfully mixed. However, GOP boundaries are content related and thus irregular. When GOP boundaries are widely spaced, dynamic following of bandwidth changes are highly inaccurate or generate delays for new users.

Summary of the Invention

Embodiments of the present invention attempt to solve the above problems by modifying the quantity of data to be injected into the channel to accord with a measured capacity or bandwidth of the channel. The quantity of data is preferably modified only by altering compression parameters, so that data fully usable in real time can be received from the channel. Thus a reduction in bandwidth of the channel leads to a drop in quality of the received signal and not to dropped frames or a reduction in performance.

According to a first aspect of the present invention there is thus a device for streaming of media file data to be sent over at least one channel having a dynamically variable bandwidth, comprising

an input for receiving media file data at a first rate,

a multiple stream creator for creating a plurality of streams of said media file data, each stream compressed to predetermined other rates, each stream comprising a series of autonomous compressed data units which are mutually compatible, and

a stream selector for dynamically matching a bandwidth of the channel to rates of said streams to send said data file as a combination of said autonomous units from said streams via said channel,

wherein said autonomous units have durations which are one of a group comprising fixed predetermined durations and durations determined by changes in said channel bandwidth.

Preferably, said stream selector is operable to select a combination of said streams in succession said combination being chosen to produce an average stream rate over a period of time which matches said channel bandwidth.

Preferably, said bandwidth of said channel is liable to vary dynamically and wherein said stream selector is operable to repeat said matching throughout said streaming operation.

Preferably, said bandwidth of said channel is liable to vary dynamically and wherein said stream selector is operable to repeat said matching throughout said streaming operation.

Preferably, said autonomous units match each other across said plurality of streams, thereby to enable switching between said streams.

Aylo Exhibit 1002
Page 392 of 1405

Preferably, said data file is arranged in said streams in a compressed media format utilizing reference frames and delta frames and wherein each autonomous unit is self-contained in respect of reference frames.

In an embodiment, said channel is a cellular telephone channel. Alternatively, said channel is an Internet channel.

Preferably, the predetermined other rate is different for each stream. In another embodiment, the predetermined other rate is identical for a set of said streams, each stream in said set being assigned a different balance between quality and speed of said media.

A preferred device comprises a plurality of media types, said predetermined other rate being identical for a set of said streams, each stream in said set being assigned a different balance between quality of said different media types.

Preferably, streams in said set are selectable by a signal received from said channel.

Preferably, streams in said set are selectable by a signal received from said channel.

An embodiment comprises a detector for detecting a change in the bandwidth of said at least one channel and operable to request a corresponding change in the data rate, and wherein the device is operable, upon receipt of said request, to set a delay, and to issue a reference frame within said delay.

The device preferably issues said reference frame at the end of said delay.

The device alternatively issues said reference frame at the occurrence of a largest detected interframe change within the time delay.

According to a second aspect of the invention there is provided a multiple stream creator for receiving a data stream at a first data rate and converting said data stream into a plurality of data streams at a plurality of data rates, said creator comprising an input for receiving said data stream at said first data rate, and a plurality of outputs, each operable to receive said data stream at said first rate and to output said data stream at a predetermined rate individually set for each output as a series of autonomous units having a duration, and wherein said duration is one of a group comprising a fixed predetermined duration and a duration dynamically changeable in accordance with bandwidth conditions in a transmission channel.

Preferably said outputs each comprise data buffers.

Aylo Exhibit 1002
Page 393 of 1405

The multiple stream creator preferably comprises a data stream processor for processing said data stream received at said first rate into said plurality of autonomous units, said autonomous units being synchronized blocks, thereby allowing said output streams to be mixed in said transmission channel to produce continuously usable output.

Preferably, said received data stream is in an uncompressed data format and said data is compressed into a compressed data format independently for each one of said output data streams.

According to a third aspect of the invention there is provided a method of sending a datastream over a variable bandwidth channel comprising modifying a quantity of data to be injected into the channel to accord with a measured capacity of the channel, said method comprising the steps of

compressing the datastream, using an algorithm having compression parameters, to produce a plurality of compressed datastreams each comprising reference frames and delta frames, each compressed datastream produced by using the algorithm together with different parameters, and

synchronizing reference frames between the compressed datastreams so that combinations of said compressed datastreams can be sent via said channel.

Preferably, said step of synchronizing reference frames is carried out at intervals determined by a dynamic status of said channel bandwidth.

Alternatively or additionally, said step of synchronizing reference frames is carried out at predetermined intervals.

Alternatively or additionally, said combination of said compressed datastreams is selected substantially dynamically to match the measured capacity of the channel.

Preferably, a change in bandwidth of a channel causes a request for commencement of a new autonomous unit, and wherein said request causes initiation of measurement of a time delay, and wherein a new autonomous unit is commenced within said time delay.

Preferably, the new autonomous unit is commenced at the end of said time delay.

Preferably, the step is provided of determining the largest delta frame during the time delay and sending a reference frame, to initiate said new autonomous unit, in place of said determined largest delta frame.

Aylo Exhibit 1002
Page 394 of 1405

According to a fourth aspect of the present invention there is provided a tool for controlling receipt of media file data over a network, wherein said media file data is available at a plurality of quality levels, the tool comprising a quality selector to enable a user to select between said different quality levels for receipt.

Preferably, said media file comprises a plurality of data groups and each quality level is a different balance between quality levels of the plurality of data groups.

Preferably, said quality selector comprises a slider operable to be moved along a scale, the position of the slider on the scale indicating a quality level.

Brief Description of the Drawings

For a better understanding of the invention and to show how the same may be carried into effect, reference will now be made, purely by way of example, to the accompanying drawings, in which:

Fig. 1 is a generalized diagram showing a data source sending data to a series of targets over a variable bandwidth network,

Fig. 2 is a generalized block diagram showing a first embodiment of the present invention in which feedback concerning available bandwidth within a channel is used to modify parameters governing compression of the datastream,

Fig. 3 is a generalized block diagram showing a second embodiment of the present invention in which a series of compressed datastreams are continually created and are selected for use in individual channels according to measured parameters of the channel,

Fig. 4 is a generalized block diagram showing the channel manager and part of a downlink manager of Fig. 3 in greater detail,

Fig. 5 is a generalized block diagram showing a third embodiment of the present invention in which the user is able to set a quality level of the data that is being sent,

Fig. 6 is a generalized schematic diagram of a quality level selection tool for use by a user in selecting a quality level in the embodiment of Fig. 5, and

Fig. 7 is a generalized schematic diagram of another quality level selection tool for use with the device of Fig. 5.

Description of the Preferred Embodiments

Reference is now made to Fig. 1, which is a generalized diagram showing a data source 10 sending data to a series of targets 12.1..12.n over a variable bandwidth network 14. The network may be a cellular telephone network or the Internet or like wide area data network or a combination. In both cases, the bandwidth of the data network is variable. In all cases the larger the number of users the lower the bandwidth per user, and radio-based networks may also suffer from reduced bandwidth due to environmental conditions. Furthermore, in the case of cellular telephone networks, events such as handover between two base stations may also affect the available bandwidth.

Data is generally compressed before sending over the channel. The level of compression may be preselected to accord with known or likely channel capabilities, as described in the introduction. However such a solution has to concentrate on the lower margins of available bandwidth and thus fails to make full use of higher bandwidth levels when available. Conversely, solutions which send datastreams at higher data rates and do not take the lower bandwidth levels into account lead to unacceptable breaks and delays in performance at the receiving end.

Reference is now made to Fig. 2 which is a generalized block diagram showing a first embodiment of the present invention in which feedback concerning available bandwidth within a channel is used to modify parameters governing compression of a datastream.

A multimedia datastream is received from a source 20 such as a host or server. The datastream may be a video datastream and may be in uncompressed or barely compressed format, for example AVI or MPEG1. The datastream is passed to a compression unit 22 which compresses the data into a very low bit-rate stream using compression parameters. A typical format that may be used for the very low bit-rate stream is MPEG 4 which uses a system based on reference frames and delta frames to compress the image part of the video data. H.263 is another typical format that may be used. A video frame is selected as a reference frame and encoded in its entirety. Then a series of subsequent frames are selected, but for each of these frames, the only data that is encoded is that which has changed with respect to the previous frame. The encoded changes form what is known as the delta frame.

Whichever of the available compression formats are used, compression is governed by a series of parameters 24. This allows the compression level to be raised

Aylo Exhibit 1002
Page 396 of 1405

or lowered depending on the data rate required and the quality of picture desired. A greater level of compression gives a lower quality of data at a lower data rate and a lower level of compression gives a higher quality of data at a higher data rate.

In the embodiment of Fig. 2 a channel capacity measurement 26 is received from a channel 28 over which the transmission occurs. The measurement may be received from a receiving client who simply reports the received data rate. If the received data rate is reported to be lower than the rate at which data is being sent then it is assumed that data is being produced too fast for the channel and the parameters 24 are altered to increase the compression level until the data output rate is the same as the channel rate. Conversely, if the data rate rises at the receiving end, the parameters 24 may be altered to reduce the compression level and increase the output data rate.

Reference is now made to Fig. 3, which is a generalized block diagram showing a second embodiment of the present invention in which a series of compressed datastreams are continually created and are selected for use in individual channels according to measured parameters of the channel. Parts that are the same as those mentioned in previous embodiments are given the same reference numerals and are not referred to again except as necessary for an understanding of the present embodiment.

In the embodiment of Fig. 3, an input datastream is received from a source 20 as before. The stream is passed to a channel manager 30, which compresses the input datastream. However, instead of compressing it once only, it compresses it into a plurality of output data streams, each one using different compression parameters so as to produce a plurality of streams each having a different data rate. The various streams are then passed to one or more downlink managers 32.1...32.n, each of which is able to manage a limited number of channels 28 for sending output data to a limited number of targets 34. Each target 34 is associated with a client 36, typically client software located at the target 34, which is able to send back, to the downlink manager 32, feedback data such as the received channel rate.

The downlink manager 32 simply obtains the received channel rate from the client 36 and mixes the various streams from the channel manager to produce an average stream rate which matches the channel rate as closely as possible. For example if the received channel rate is 39.5 kbits/sec and the downlink manager has available a 35 kbits/sec stream and a 45 kbits/sec stream then the downlink manager

Aylo Exhibit 1002
Page 397 of 1405

will attempt to mix the two streams in suitable proportions to match the 39.5 kbits/sec channel. Thus the target 36 may receive a continuous stream of data, the quality of the channel influencing only the quality of the data and not the continuity.

It will be recalled that the compression systems in use include reference and delta frame style compression. Such compression systems, when used to produce streams at different data rates, do not produce delta frames which are interchangeable between the different data rates. Thus in order to create streams that are interchangeable, it is preferable to arrange all of the streams as a series of self-contained packets of given time duration. Each of the packets is self-contained in that it begins with a reference frame and hence does not depend on previously transmitted data. Thus the target is able to use consecutive packets from different streams without any data discontinuity.

The self-contained units are preferably of relatively short duration. The shorter the duration the better the system is able to follow dynamically changes in the bandwidth of the connection. On the other hand, since each unit must start with a reference or key frame, and such frames are very costly in bandwidth since they must be transmitted in full, the use of short self-contained units itself increases the required bandwidth. A compromise unit length of less than ten seconds, preferably 3, 5 or 7 seconds, is preferred.

In an alternative embodiment the length of the self-contained unit is itself allowed to vary dynamically in accordance with circumstances. When the downlink manager 32 detects a significant change in the channel bandwidth it issues a request for a new data rate from the channel manager 30. Alternatively, a new target may be requesting the data feed. The channel manager 30 initiates a time delay when a first request is received, the delay being less than ten seconds, as discussed above, and at the end of the delay starts new self-contained units at all data rates. Thus dynamic following of changes in bandwidth is provided whilst transmitting the minimum possible number of key or reference frames. If no request is received, then the system continues to use delta frames as long as it is practical.

In a variation of the above, instead of waiting to the end of the time delay, the channel manager first of all looks ahead at the approaching data stream and checks whether there is a reference frame approaching during the time delay period in any case. If so, no further action is needed. If there is no reference frame then the channel manager looks ahead to find the biggest delta frame, representing the biggest

Aylo Exhibit 1002
Page 398 of 1405

instantaneous change in the picture, over the delay time period. A reference frame is then inserted in place of the identified largest delta frame. Thus, the insertion of a full reference frame is optimized to the data.

Reference is now made to Fig. 4 which is a generalized block diagram showing the channel manager 30 and a part of a downlink manager 32 of Fig. 3 in greater detail. Parts that are the same as those mentioned in previous embodiments are given the same reference numerals and are not referred to again except as necessary for an understanding of the present embodiment.

As shown in Fig. 4, the channel manager 30 comprises a series of compression units 40.1, 40.2, 40.3, etc. Each of these compression units is preferably operative to compress an input signal received from source 20, using a different compression parameter or set of compression parameters to produce an output having a different data rate. Each compression unit is followed by a buffer 41. The buffer 41 may be used to arranged the compressed data into packets of fixed time duration.

The downlink manager 32 comprises a series of selector switches 42, one for each channel being managed. For the sake of simplicity, only one such switch 42 is shown in the figure. The switch 42 receives each one of the data streams produced by the source 20 and selects one of them at any given time to be sent to the channel 28. Selection is carried out based on a channel capacity measurement 44, which is received from the client 36.

Reference is now made to Fig. 5, which is a generalized block diagram showing a third embodiment of the present invention in which the user is able to set a quality level of the data that is being sent. Parts that are the same as those mentioned in previous embodiments are given the same reference numerals and are not referred to again except as necessary for an understanding of the present embodiment. As described above, compression parameters are altered in order to change the data rate. Generally speaking, there are numerous parameters that can be set in order to alter the data rate, and there are numerous combinations of these parameters that can give the same data rate. For example, parameters that may be altered include:

the extent of image compression within an individual frame,

the number of frames per second to be transmitted,

the proportion of reference frames to delta frames, and

a balance between compression levels of different parts of the data – for example video data includes both sound and image. It is possible to increase a level

Aylo Exhibit 1002
Page 399 of 1405

of compression to the image and to decrease a level of compression to the sound or vice versa.

In Fig. 5, part of a channel manager 30 is shown as well as a switch 42 of a downlink manager. In the channel manager of the embodiment of Fig. 5, a series of data compressors 50 are provided for each required data rate. The data compressors are labeled high, medium and low, although in fact they are different compromises between compression parameters. For example "high" may indicate high image quality but low frame rate, whereas "low" may indicate low image quality but high frame rate. The selector switch 48 is the same as in the previous embodiment except that it now chooses channels based on two different items of information. One item of information is the received channel rate 44, which selects the data rate as before. The other item of information is a data quality request which is received from the target. This selects, within the given data rate, between the different quality levels.

Reference is now made to Fig. 6, which is a generalized schematic diagram of a quality level selection tool 60 for use by a user in selecting a quality level in the embodiment of Fig. 5. The selection tool 60 comprises a simple longitudinal bar 62 which may be marked with labels and numbers, and a slider 64 for moving along the bar 62. The bar shown in Fig, 6 indicates that the user may select different compromises between sound quality and image quality.

Reference is now made to Fig. 7, which is a generalized schematic diagram of another quality level selection tool 70 for use by a user in selecting a quality level in the embodiment of Fig. 5. The selection tool 70 is likewise in the form of a bar and slider but this time the choice is between high quality images at a low frame rate (marked "still") and low quality images at a high frame rate (marked "motion"). It will be appreciated that a low bandwidth "high" may yield a lower image quality than a high bandwidth "low" and vice versa.

Embodiments of the present invention thus provide a system and method in which the measured received data level of a channel is used to select a transmitted data rate so that full advantage may be taken of fluctuating channel capacity levels to send the highest possible quality multimedia data.

It is appreciated that certain features of the invention, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the invention

Aylo Exhibit 1002
Page 400 of 1405

which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable subcombination.

It will be appreciated by persons skilled in the art that the present invention is not limited to what has been particularly shown and described hereinabove. Rather the scope of the present invention is defined by the appended claims and includes both combinations and subcombinations of the various features described hereinabove as well as variations and modifications thereof which would occur to persons skilled in the art upon reading the foregoing description.

Claims

1. A device for streaming of media file data to be sent over at least one channel having a dynamically variable bandwidth, comprising

an input for receiving media file data at a first rate,

a multiple stream creator for creating a plurality of streams of said media file data, each stream compressed to predetermined other rates, each stream comprising a series of autonomous compressed data units which are mutually compatible, and

a stream selector for dynamically matching a bandwidth of the channel to rates of said streams to send said data file as a combination of said autonomous units from said streams via said channel,

wherein said autonomous units have durations which are one of a group comprising fixed predetermined durations and durations determined by changes in said channel bandwidth.

2. A device according to claim 1, wherein said stream selector is operable to select a combination of said streams in succession said combination being chosen to produce an average stream rate over a period of time which matches said channel bandwidth.

3. A device according to claim 1, wherein said bandwidth of said channel is liable to vary dynamically and wherein said stream selector is operable to repeat said matching throughout said streaming operation.

4. A device according to claim 2, wherein said bandwidth of said channel is liable to vary dynamically and wherein said stream selector is operable to repeat said matching throughout said streaming operation.

5. A device according to claim 1, wherein said autonomous units match each other across said plurality of streams, thereby to enable switching between said streams.

6. A device according to claim 5, wherein said data file is arranged in said streams in a compressed media format utilizing reference frames and

delta frames and wherein each autonomous unit is self-contained in respect of reference frames.

7. A device according to claim 1, wherein said channel is a cellular telephone channel.

8. A device according to claim 1, wherein said channel is an Internet channel.

9. A device according to claim 1, said predetermined other rate being different for each stream.

10. A device according to claim 1, said predetermined other rate being identical for a set of said streams, each stream in said set being assigned a different balance between quality and speed of said media.

11. A device according to claim 1, said media file comprising a plurality of media types, said predetermined other rate being identical for a set of said streams, each stream in said set being assigned a different balance between quality of said different media types.

12. A device according to claim 10, said streams in said set being selectable by a signal received from said channel.

13. A device according to claim 11, said streams in said set being selectable by a signal received from said channel.

14. A device according to claim 1, comprising a detector for detecting a change in the bandwidth of said at least one channel and operable to request a corresponding change in the data rate, and wherein the device is operable, upon receipt of said request, to set a delay, and to issue a reference frame within said delay.

15. A device according to claim 14, operable to issue said reference frame at the end of said delay.

16. A device according to claim 14, operable to issue said reference frame at the occurrence of a largest detected interframe change within the time delay.

17. A multiple stream creator for receiving a data stream at a first data rate and converting said data stream into a plurality of data streams at a plurality of data rates, said creator comprising an input for receiving said data stream at said first data rate, and a plurality of outputs, each operable to receive said data stream at said first rate and to output said data stream at a predetermined rate individually set for each output as a series of autonomous units having a duration, and wherein said duration is one of a group comprising a fixed predetermined duration and a duration dynamically changeable in accordance with bandwidth conditions in a transmission channel.

18. A multiple stream creator according to claim 17, wherein said outputs each comprise data buffers.

19. A multiple stream creator according to claim 17, comprising a data stream processor for processing said data stream received at said first rate into said plurality of autonomous units, said autonomous units being synchronized blocks, thereby allowing said output streams to be mixed in said transmission channel to produce continuously usable output.

20. A multiple stream creator according to claim 17, wherein said received data stream is in an uncompressed data format and wherein said data is compressed into a compressed data format independently for each one of said output data streams.

21. A method of sending a datastream over a variable bandwidth channel comprising modifying a quantity of data to be injected into the channel to accord with a measured capacity of the channel, said method comprising the steps of

compressing the datastream, using an algorithm having compression parameters, to produce a plurality of compressed datastreams each comprising reference frames and delta frames, each compressed datastream produced by using the algorithm together with different parameters, and

synchronizing reference frames between the compressed datastreams so that combinations of said compressed datastreams can be sent via said channel.

22. A method according to claim 21, wherein said step of synchronizing reference frames is carried out at intervals determined by a dynamic status of said channel bandwidth.

23. A method according to claim 21, wherein said step of synchronizing reference frames is carried out at predetermined intervals.

24. A method according to claim 21, wherein said combination of said compressed datastreams is selected substantially dynamically to match the measured capacity of the channel.

25. A method according to claim 22, wherein a change in bandwidth of a channel causes a request for commencement of a new autonomous unit, and wherein said request causes initiation of measurement of a time delay, and wherein a new autonomous unit is commenced within said time delay.

26. A method according to claim 25, wherein said new autonomous unit is commenced at the end of said time delay.

27. A method according to claim 25, comprising the step of determining the largest delta frame during the time delay and sending a reference frame, to initiate said new autonomous unit, in place of said determined largest delta frame.

28. A tool for controlling receipt of media file data over a network, wherein said media file data is available at a plurality of quality levels, the

Aylo Exhibit 1002
Page 405 of 1405

tool comprising a quality selector to enable a user to select between said different quality levels for receipt.

29. A tool according to claim 28, wherein said media file comprises a plurality of data groups and wherein each quality level is a different balance between quality levels of the plurality of data groups.

30. A tool according to claim 28, wherein said quality selector comprises a slider operable to be moved along a scale, the position of the slider on the scale indicating a quality level.

31. A device for streaming of media file data to be sent over at least one channel having a dynamically variable bandwidth, substantially as hereinbefore described with reference to the accompanying drawings.

32. A multiple stream creator for receiving a data stream at a first data rate and converting said data stream into a plurality of data streams at a plurality of data rates, substantially as hereinbefore described with reference to the accompanying drawings.

33. A method of sending a datastream over a variable bandwidth channel comprising modifying a quantity of data to be injected into the channel to accord with a measured capacity of the channel, substantially as hereinbefore described with reference to the accompanying drawings.

34. A tool for controlling receipt of media file data over a network, substantially as hereinbefore described with reference to the accompanying drawings.

Aylo Exhibit 1002
Page 406 of 1405

INVESTOR IN PEOPLE

Application No: GB 0023062.3 **Examiner:** John Cullen
Claims searched: 1-27, 31-33 **Date of search:** 6 November 2001

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.S): H4P (PDCFM, PDCFP, PDCFX, PF)

Int Cl (Ed.7): H03M 7/30; H04L 1/00; H04N 7/26, 7/32

Other: Online: WPI, EPODOC, JAPIO

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
A, E	EP 1049074 A1	(LUCENT) See Abstract, Figs. 2 and 4, and paragraphs 14-18 and 28-35.	---
A, E	WO 00/72601 A2	(BIGBAND NETWORKS) See Figs. 1-4, 6, 7 and 9, line 26 of p15 to line 24 of p17 and line 10 of p21 to line 2 of p22.	---
A	WO 99/00984 A1	(CITRIX) See Fig. 15 and line 13 of p30 to line 14 of p33.	---
A	WO 98/09387 A1	(ERICSSON) See Figs. 3A, 6 and 7, and line 35 of p14 to line 26 of p15.	---
A	US 5847760	(OPTIBASE) See whole document.	---

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.

(19) Canadian Intellectual Property Office
An Agency of Industry Canada

Office de la Propri,t, Intellectuelle du Canada
Un organisme d'Industrie Canada

(11) CA 2 466 482 (13) A1

(40) 22.05.2003
(43) 22.05.2003

(12)

(21) 2 466 482

(22) 09.11.2002

(51) Int. Cl.[7]: H04J 3/16, H03M 7/00, G06F 12/02, H04N 5/92

(85) 07.05.2004

(86) PCT/US02/035903

(87) WO03/042783

(30) 60/337,939 US 09.11.2001
10/291,210 US 08.11.2002

(71) MUSICMATCH, INC.,
270-16935 West Bernardino Drive, SAN DIEGO , XX (US).

(72) DORON, AMIR (US).
DUNNING, TED E. (US).
GRUHNE, MATTHIAS (US).

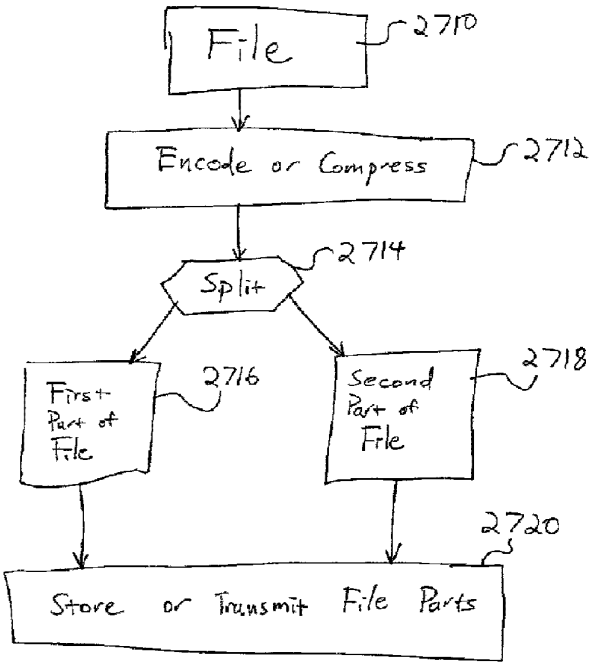
(74) SIM & MCBURNEY

(54) DECOUPAGE DE FICHIERS, CODAGE ECHELONNABLE, ET TRANSMISSION ASYNCHRONE DANS UN TRANSFERT CONTINU DE DONNEES

(54) FILE SPLITTING SCALADE CODING AND ASYNCHRONOUS TRANSMISSION IN STREAMED DATA TRANSFER

(57)

Files (2750) are divided into parts (2716, 2718) and at least some of the parts are transmitted to a client using a communication channel. At least some of the transmitted parts are cached locally. This allows subsequent streaming playback of the file while using 'less bandwidth by transmitting the part of the file that hasn't been cached, and combining the cached parts with the tramitted parts. In some embodiments, files may be represented at a low quality level by a first data set (2804), and at higher quality levels with additional data sets (2806). Data sets are cached locally, so that during subsequent streaming playback of the file, the quality level of the playback may be improved by sending additional data sets using bandwidth that would otherwise be dedicated to transmitting the cached data sets.



Office de la Propriété Intellectuelle du Canada
Un organisme d'Industrie Canada

Canadian Intellectual Property Office
An agency of Industry Canada

CA 2466482 A1 2003/05/22

(21) 2 466 482

(12) DEMANDE DE BREVET CANADIEN
CANADIAN PATENT APPLICATION

(13) A1

(86) Date de dépôt PCT/PCT Filing Date: 2002/11/09
(87) Date publication PCT/PCT Publication Date: 2003/05/22
(85) Entrée phase nationale/National Entry: 2004/05/07
(86) N° demande PCT/PCT Application No.: US 2002/035903
(87) N° publication PCT/PCT Publication No.: 2003/042783
(30) Priorités/Priorities: 2001/11/09 (60/337,939) US; 2002/11/08 (10/291,210) US

(51) Cl.Int.^{7}/Int.Cl.^{7} H04J 3/16, G06F 12/02, H04N 5/92, H03M 7/00

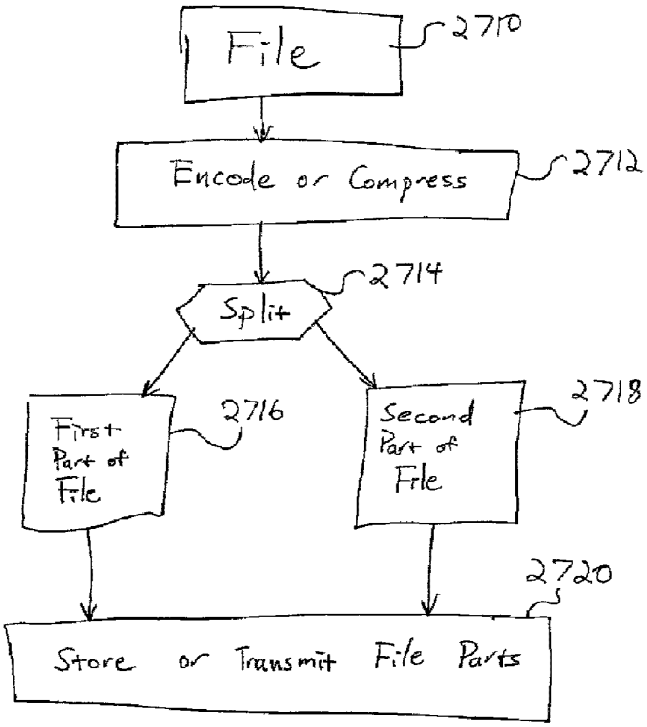
(71) Demandeur/Applicant:
MUSICMATCH, INC., US

(72) Inventeurs/Inventors:
DUNNING, TED E., US;
DORON, AMIR, US;
GRUHNE, MATTHIAS, US

(74) Agent: SIM & MCBURNEY

(54) Titre : DECOUPAGE DE FICHIERS, CODAGE ECHELONNABLE, ET TRANSMISSION ASYNCHRONE DANS UN TRANSFERT CONTINU DE DONNEES

(54) Title: FILE SPLITTING SCALADE CODING AND ASYNCHRONOUS TRANSMISSION IN STREAMED DATA TRANSFER



(57) Abrégé/Abstract:
Files (2750) are divided into parts (2716, 2718) and at least some of the parts are transmitted to a client using a communication channel. At least some of the transmitted parts are cached locally. This allows subsequent streaming playback of the file while using 'less bandwidth by transmitting the part of the file that hasn't been cached, and combining the cached parts with the

(21) 2 466 482

(13) A1

(57) Abrégé(suite)/Abstract(continued):
tramitted parts. In some embodiments, files may be represented at a low quality level by a first data set (2804), and at higher quality levels with additional data sets (2806). Data sets are cached locally, so that during subsequent streaming playback of the file, the quality level of the playback may be improved by sending additional data sets using bandwidth that would otherwise be dedicated to transmitting the cached data sets.

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
22 May 2003 (22.05.2003)

PCT

(10) International Publication Number
WO 03/042783 A3

(51) International Patent Classification[7]: H04J 3/16, G06F 12/02, H04N 5/92, H03M 7/00

(21) International Application Number: PCT/US02/35903

(22) International Filing Date:
9 November 2002 (09.11.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
60/337,939 9 November 2001 (09.11.2001) US
Not furnished 8 November 2002 (08.11.2002) US

(71) Applicant: **MUSICMATCH, INC.** [US/US]; 1635 W. Bernardino Drive, Suite 270, San Diego, CA 92127 (US).

(72) Inventors: **DUNNING, Ted, E.**; 11797 River Rim Road, San Diego, CA 92126 (US). **DORON, Amir**; 8455 Lake Ben Avenue, San Diego, CA 92119 (US). **GRUHNE, Matthias**; Schulstrasse 2a, 99718 Westgreussen (US).

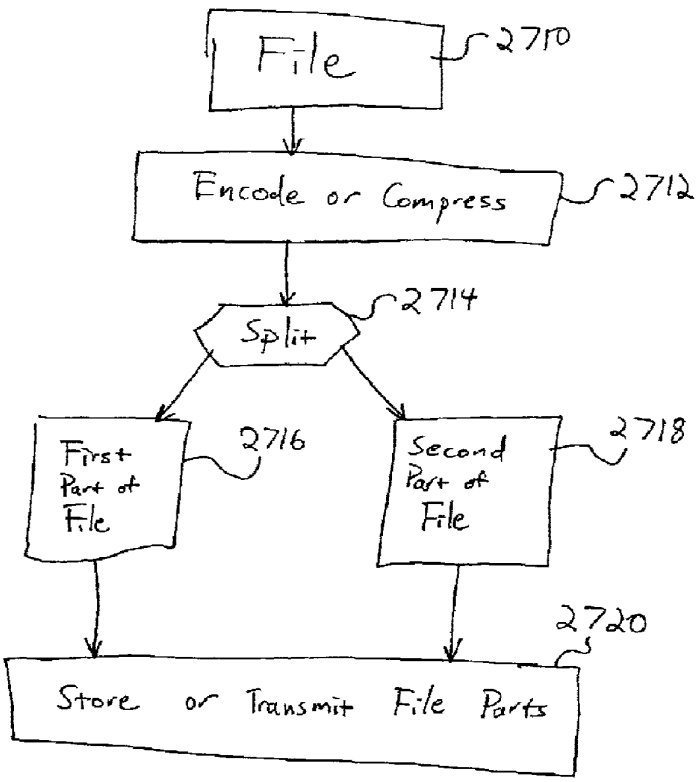
(74) Agents: **PLIMIER, Michael, D.** et al.; Fenwick & West LLP, Two Palo Alto Square, Palo alto, CA 94306 (US).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, 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, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN, YU, ZA, ZM, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK,

[Continued on next page]

(54) Title: FILE SPLITTING SCALADE CODING AND ASYNCHRONOUS TRANSMISSION IN STREAMED DATA TRANSFER



(57) Abstract: Files (2750) are divided into parts (2716, 2718) and at least some of the parts are transmitted to a client using a communication channel. At least some of the transmitted parts are cached locally. This allows subsequent streaming playback of the file while using 'less bandwidth by transmitting the part of the file that hasn't been cached, and combining the cached parts with the tramitted parts. In some embodiments, files may be represented at a low quality level by a first data set (2804), and at higher quality levels with additional data sets (2806). Data sets are cached locally, so that during subsequent streaming playback of the file, the quality level of the playback may be improved by sending additional data sets using bandwidth that would otherwise be dedicated to transmitting the cached data sets.

WO 03/042783 A3

WO 03/042783 A3



TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

(88) Date of publication of the international search report:
10 July 2003

Published:
— *with international search report*
— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

FILE SPLITTING, SCALABLE CODING, AND ASYNCHRONOUS TRANSMISSION IN STREAMED DATA TRANSFER

Cross-Reference to Related Application

[0001] This application is a continuation-in-part of U.S. Patent Application Serial Number 09/846,823, filed on April 30, 2001, Patent No. X,XXX,XXX, the disclosure of which is incorporated by reference, and which claimed priority from U.S. Provisional Application Serial Number 60/201,622, filed May 3, 2000, the disclosure of which is incorporated by reference. The present application also claims priority from provisional U.S. Patent Application Serial No. 60/337,939, for "File Splitting, Scalable Coding, and Asynchronous Transmission in Streamed Data Transfer," filed November 9, 2001, the disclosure of which is incorporated by reference.

Background of the Invention

Field of the Invention

[0002] The present invention is related to delivery of streamed programs such as audio and video, and more particularly to systems and methods providing improved control, efficiency, and quality of such streamed transmissions using file splitting, scalable coding, and asynchronous transmission.

Description of the Background Art

[0003] Delivery of audio and video programs over the Internet provides several advantages over conventional broadcast media such as radio and television. Unlike broadcast media, which require users to receive programs at particular times, or to record them for later use, Internet delivery allows users to select and receive programs upon demand, at a time that is convenient to them. For example, a user may browse to a news story, presented in text format, and click on a link that initiates playback of an audio report on the news item. Users may also click on links to hear songs or song samples, as is common in e-commerce sites, such as amazon.com or cdnow.com, in order to permit users to sample songs before purchasing compact discs (CDs). Internet delivery of audio programming may also be used for implementing personalized radio stations, which deliver music tracks selected in response to the tastes of particular users. Similar functionality is available for the delivery of video programs on demand, allowing users to view sports highlights, news reports, music videos, and even films and television shows, over the Internet.

[0004] Software for playing back such audio and video files is known in the art, including for example the Windows Media Player from Microsoft Corporation, and the RealPlayer from Real Networks, Inc. Functionality for audio and video delivery may also be built into browsers such as Netscape Navigator and Microsoft Internet Explorer.

[0005] One disadvantage of Internet delivery of audio and video programs is the relatively limited bandwidth that is available for any particular program. Compared with broadcast media, such as radio, television (over-the-air or cable), and the like, typical Internet connections offer far less bandwidth and thus limit the amount of information that may be transmitted to the end user in a given period of time. Thus, even with the application of aggressive compression algorithms, sound quality and picture quality of streamed programs received over the Internet tend to be substandard. For example, due to the limited number of bits available, existing Internet delivery schemes fail to provide received audio programs at a quality approaching that of CDs.

[0006] As an alternative to streamed programs, and in order to improve quality of the received programs, users may download programs and then, subsequent to downloading them, play the received program via their computers. Since the program is not being viewed or listened to in "real time," more time is available to transmit and receive the program, so that larger files may be provided, and better quality achieved. However, it may take many minutes, or even hours, to download a short program at a reasonable level of quality. Thus, such a scheme does not provide the user with an immediate listening or viewing experience; rather, the user must wait until the program is downloaded before it can be viewed or listened to.

Summary of the Invention

[0007] The present invention provides improved quality, security, and efficiency for streamed programs, by combining received streamed data with previously-stored (cached) data. In one embodiment, a program file containing program data such as audio, video, or the like, is split into two or more sections. In one embodiment, each section is not usable on its own to provide program output (e.g., audio signals where the program file is an audio file), but can only be used when recombined with the remaining sections. In one embodiment, one section of the split data is considerably smaller than the other section(s). This splitting allows the second section to be cached by a client machine without a danger that the client has a persistent usable copy of the original file. This allows the benefits of caching to be realized without providing users with unauthorized persistent copies of files.

[0008] For example, the second section may be transmitted in advance of the actual playback of a program file, thus avoiding bottlenecks that may result when the entire program file is transmitted upon user demand. Until the first section is transmitted, the user will not be able to play the program file. The smaller first section may be streamed to allow playback of the program file. The first section is discarded after playback so that no persistent usable copy of the program file remains on the user's machine. The second section remains cached, allowing future streaming playback to be performed efficiently. The ability to not provide a persistent usable

copy of the program file can be advantageous because copyright holders may not wish to authorize such persistent copies, since the persistent copy could cannibalize revenues from sales of the program file in a tangible medium (such as audio CDs), or paid future downloads, streamed or otherwise, of the program file.

[0009] Alternatively, the program file is split into sections in order to provide scalability in the playback quality level. The first section is playable at a relatively low quality level. Additional sections of data may be added to the first section to increase the quality level. Thus, the program file may be downloaded and played at a quality level determined by the bandwidth of the connection. The sections may be cached so that when it is desired to play the program file again, additional sections may be downloaded and added for higher quality playback.

[0010] In some embodiments, a persistent copy of a playable version of the program file is not stored on the client. The additional sections are not playable without the first level. Thus, the additional sections may be cached, so that the file can be played back by combining a downloaded first section with cached additional sections.

[0011] The file splitting and scalable coding embodiments may also both be used on the same program file. This allows most of the file to be cached while still providing the server control over when the file may be played back. It also allows subsequent playbacks of the file after the first to be progressively higher quality as more of the scalable sections are transmitted with every playback.

Brief Description of the Drawings

[0012] Fig. 1A is a block diagram of a functional architecture.

[0013] Fig. 2 is a block diagram of a conceptual architecture for one embodiment.

[0014] Figs. 3A and 3B illustrate a method for splitting a program file according to one embodiment of the present invention.

[0015] Figs. 3C and 3D illustrate a method of receiving and playing a previously split program file according to one embodiment of the present invention.

[0016] Fig. 4A illustrates an example of a scalably-coded program file.

[0017] Fig. 4B is a block diagram illustrating the use and combination of primary data and secondary data to provide different playback quality levels.

[0018] Figs. 4C and 4D depict examples of scalable coding according to one embodiment of the present invention.

[0019] Figs. 5A and 5B depict techniques for combining file splitting and scalable coding according to one embodiment of the present invention .

Detailed Description of the Preferred Embodiments

[0020] The following description of preferred embodiments of the present invention is presented in the context of an Internet-based jukebox or personalized radio station where it is desirable to stream music files. One skilled in the art will recognize that the present invention may be implemented in many other domains and environments, both within the context of musical files, and in other contexts. Accordingly, the following description, while intended to be illustrative of a particular implementation, is not intended to limit the scope of the present invention or its applicability to other domains and environments. Rather, the scope of the present invention is limited and defined solely by the claims.

Architecture Overview

[0021] Referring now to Fig. 2, there is shown a conceptual architecture of one embodiment of the present invention. In the architecture of Fig. 2, the invention is implemented in connection with a web-based "jukebox" 103, or personalized radio station.

[0022] Stream delivery system 150 interacts with jukebox 103 to specify a sequence of audio files to deliver to jukebox 103. In some embodiments, this jukebox 103 may run on a client such as a personal computer, while the stream deliver server system is part of a remote server, connected to the client via a network. Jukebox 103 transmits requests to stream delivery system 150, and stream delivery system 150 delivers the audio files, as tracks, to jukebox 103. Stream delivery system 150 also communicates with real-time subscription authorization module 157, which includes real-

time server 154 and database 156 that keep track of which user accounts are active and enforces global business rules about which accounts can listen to the radio at a given time. Within stream delivery system 150, there are a number of distinct software entities. Radio sequence generator 1613 receives requests from jukebox 103, receives format definitions 1611 and general constraints 1616, and receives recommendations from recommendation engine 107, to generate track selections to be transmitted to jukebox 103. The track selections generated by radio sequence generator 1613 specify which files to play according to estimated listener preferences as well as predetermined station formats. Authorization and content server 1614 keeps a record of the files that are selected by radio sequence generator 1613; server 1614 is consulted by radio sequence generator 1613 when files are requested. If generator 1613 does not provide the necessary security information, server 1614 flags this anomaly and declines to provide the data.

[0023] Referring now to Fig. 1A, there is shown a block diagram of a functional architecture for one embodiment of the present invention. Content index 110 provides a concise index of content stored in database 102, and is generated by conventional index generation means, to enable more efficient searching and updating of database 102.

Stream Delivery

[0024] In one embodiment, program files, such as audio files, are delivered to users as streamed data. The relationship discovery engine described in U.S. Patent Application Serial Number 09/846,823, filed on April 30, 2001, Patent No. X,XXX,XXX, describes one way to implement a personalized radio station, which may be used in conjunction with the present invention. The radio sequence transmitter 121 may deliver units of data to jukebox 103 in a format wherein each unit encodes a period of music. Since radio stations typically repeat their programming several times, it is beneficial to cache the data units in order to reduce the amount of transmitted data. In addition, if a sufficiently large time scale is used, different channels of the radio station may have considerable overlap among currently playing selections that are being delivered to various users. By identifying these common units, trans-

mitter 121 can take advantage of further economies of transmission, so as to provide more efficient delivery of audio data.

[0025] In one embodiment, the transmitter 121 employs file splitting to improve the streaming performance, and provide streaming performance even in situations where channel capacity would otherwise be insufficient to provide the desired level of quality. In some embodiments, files are split prior to being stored in the content database 102 or with the compressed signal files 1615. In other embodiments, the files 101 are stored in the content database 102 or with the compressed signal files 1615 without first being split; the files are then split prior to being transmitted to a user for playback.

[0026] Fig. 3A is a flow diagram illustrating a method of splitting and storing a program file (such as a bitstream) containing program data such as audio, video, or the like, according to one embodiment. The process begins with the unsplit file 2710. Prior to being split, the file 2710 may be encoded and/or compressed 2712, although such operations are not essential to the present invention. Next the file 2710 is split 2714 into two or more parts, resulting in at least a first part 2716 of the file and a second part 2718 of the file. Particular techniques for splitting 2714 file 2710 are described in more detail in connection with Fig. 3B below. In one embodiment, the second part 2718 is substantially larger than the first part 2716. The file may be split 2714 into more than two parts. The parts 2716 and 2718 of the file 2710 are then stored 2720 in the content database 102 or with the compressed signal files 1615, or in some other location for later transmission. Alternatively, in an embodiment wherein files are split 2714 immediately prior to being transmitted, parts 2716 and 2718 are transmitted 2720 to an end user or remote computer for playback.

[0027] Fig. 3B is a block diagram illustrating an example of a method of splitting 2714 file 2710 into parts 2716, 2718 according to one embodiment. In the illustrated example, the file 2710 has M bits 2722-1 through 2722-M. The system generates first part 2716 of the file 2710 by taking N out of every P bits. The system generates second part 2718 of the file by taking the remaining P-N out of every P bits. In Fig. 3B, N is equal to one, and P is equal to four. Thus, the system assigns first bit 2722-1 to first part 2716, and assigns the next three bits 2722-2 through 2722-4 to sec-

ond part 2718. The system then assigns fifth bit 2722-5 to first part 2716 and the next three bits 2722-6 through 2722-8 to second part 2718. The system repeats this pattern until all the bits of the file 2710 have been assigned to one of first and second parts 2716, 2718. In short, then, the system divides file 2710 into sections, with the first bit (or some other bit) of each section being sent to the first part 2716, and the remaining bits of each section being sent to the second part 2718. In general, the splitting technique of the present invention results in two or more sections 2716, 2718 each of which cannot be played back on its own, but can be played only when recombined with the other section or sections.

[0028] In one embodiment, N is equal to one, and P is equal to twenty, so that the first part 2716 is approximately 5% the size of the original file 2710, and the second part 2718 is approximately 95% the size of the original file 2710.

[0029] One skilled in the art will recognize that the system of the invention is not limited to file-splitting by bits, but may split files according to other units, such as bytes, or simply an arbitrary unit size. Alternatively, other file-splitting techniques may be used. For example, the system can include all headers, compression table selectors and scale factors into the first section 2716, and include all encoded program data (such as audio or video) into the second part 2718. Other methods of splitting the file 2710 into two or more parts may also be used.

[0030] In general, it is desirable in most embodiments that each part of the file 2710 is not decodable or able to be played back on its own, but can only be decoded and/or played back when recombined with the remaining part(s). This allows a part to be handled, transmitted, cached, or otherwise processed, without concerns that a user may permanently store and reuse an unauthorized copy of the program. Since a part cannot be used independently to recreate a recognizable facsimile of the signal in the original file 2710, the part can be cached by a client machine without any danger that the client has been given an unauthorized, useable persistent copy of the program file 2710. The ability to not provide a persistent usable copy of the program file can be advantageous because copyright holders may not wish to authorize such persistent copies, since the persistent copy could cannibalize revenues from sales of

the program file in a tangible medium (such as audio CDs), or paid future downloads, streamed or otherwise, of the program file.

[0031] Fig. 3C is a flow diagram illustrating a method of receiving and playing a previously split program file according to one embodiment of the present invention. When a file is to be transmitted from stream delivery system 150 to the jukebox 103, the jukebox 103 or server 1614 determines 2724 whether the second part 2718 of the file 2710 has previously been stored, or cached, locally and is available for retrieval. If the second part 2718 is not available locally for retrieval, jukebox 103requests transmission of both parts 2716, 2718 of the file 2710. In response, server 1614 retrieves the requested parts 2716, 2718 from the files 1615 or the content database 102 (depending on where the parts 2716, 2718 have been stored), and transmits both the parts 2716, 2718 of the file 2710 to the jukebox 103, which receives 2726 them. Alternatively, if the files 2710 are stored unsplit, server 1614 retrieves the file 2710 to be sent, splits the requested file according to techniques described above, and then transmits the parts 2716, 2718 to the jukebox 103, which receives 2726 them.

[0032] Upon receiving 2726 parts 2716, 2718, in one embodiment the jukebox 103 stores 2732 the second part 2718 locally, so as to avoid the need to retransmit the second part 2718 if it is needed in the future. The jukebox 103 combines 2728 both parts 2716, 2718, and plays back 2730 the recombined program file 2710. This reception 2726, combination 2728, and playback 2730 may all take place concurrently, for example according to a streamed transmission methodology. In one embodiment, as the bits arrive at the jukebox 103, they are combined 2728 and played back 2730; the jukebox 103 need not wait until the entirety of the parts 2716, 2718 are received 2726 before combining 2728 and playing them back 2730. In one embodiment, the recombined file 2710 and the first part 2716 are discarded 2731 after or during playback. Since the cached second part 2718 represents only a portion of the program file 2710 and in one embodiment cannot be decoded or played back on its own, the invention prevents users from retaining an unauthorized playable copy of the program file 2710. Ultimate control over the user's playback of the program file 2710 thus remains with the server 1614, since the stored second part 2718 cannot be used to play back of the file 2710 until the server 1614 retransmits the first part 2716.

[0033] If the second part 2718 has been received and stored locally in a prior session, jukebox 103 requests transmission of only the first part 2716 of the file 2710. In response, server 1614 retrieves the requested first part 2716 from the files 1615 or the content database 102, and transmits the first part 2716 to the jukebox 103, which receives 2734 it. Alternatively, if the files 2710 are stored unsplit, server 1614 retrieves the file 2710 to be sent, splits the requested file according to techniques described above, and then transmits the first part 2716 of the file 2710 to the jukebox 103, which receives 2734 it. The jukebox 103 retrieves 2736 the second part 2718 from local storage and combines 2738 the first and second parts 2716, 2718 to reconstruct the file 2710. The jukebox 103 plays back 2740 the reconstructed program file 2710. In one embodiment, as the bits arrive at the jukebox 103, they are combined 2738 with the stored second part 2718 to form a representation of the program file and played back 2740; the jukebox 103 need not wait until the entirety of the first part 2716 is received 2734 before combining and playing back the program. In other words, playback 2740 may take place while the first part 2716 is being streamed. As described above, in one embodiment, the recombined file 2710 and the first part 2716 are discarded 2731 after or during playback.

[0034] As can be seen from the description above, it is not necessary to transmit the second part 2718 more than once. The second part 2718 can be stored locally after the first transmission, so that, in one embodiment, only the first part 2716 need be transmitted for subsequent streaming playback of the program file 2710. Since in some embodiments the stored second part 2718 is substantially larger than the first part 2716, the present invention greatly reduces bandwidth requirements for streaming playback of the program file 2710. After the second part has been stored locally 2732, even clients with slow connections may effectively stream the program file 2710. Thus, from the user's perspective, the program file 2710 appears to be streamed in a manner equivalent to conventional streaming.

[0035] In addition, the described file splitting may provide advantages in licensing distribution of programs for playback by users. The file splitting benefits the licensor because the user does not receive a persistent usable copy of the program file, and the server retains control of playback of the program file 2710. This is advan-

tageous because in contexts where blanket licensing or compulsory licensing is available for providing streamed data, in audio programming for example, a persistent copy would cannibalize revenues from future licensed streamed data downloads, and lack of control would make collecting licensing revenues difficult. The file splitting benefits users who do not want to wait while a large file is transmitted and received before accessing the program file. By splitting the file and storing part locally, transmission of large and unwieldy amounts of data in real-time is unnecessary. The user has quick access to the program file, even if he or she is using a slow communication channel.

[0036] The present invention thus improves efficiency in the transmission of program files such as audio or video files, and helps avoids bottlenecks associated with conventional streaming techniques, while still providing control and security.

[0037] Fig. 3D is a flow diagram illustrating a method of receiving and playing a previously split program file according to one embodiment of the present invention, so as to provide improved streaming performance from a user's perspective, even if adequate bandwidth to stream the entire file 2710 is not available, through advance downloading of the second section 2718. For example, the second section 2718 may be transmitted in advance of the actual playback of an audio or video program in a first session, thus avoiding potential bottlenecks that may result when the entire program file 2710 is transmitted upon user demand according to streamed techniques. Advance transmission can be performed at off-peak times, thus avoiding bandwidth bottlenecks. This advance transmission can occur when the user is not actually listening to music, so as to facilitate improved usage of an otherwise idle network connection.

[0038] The jukebox 103 determines 2742 which program files are likely to be requested by a user, so that at idle times it can transfer data that is likely to be useful in the future. Such determination may be made, for example, by using the learned artist relationships described in U.S. Patent Application Serial Number 09/846,823, filed on April 30, 2001, Patent No. X,XXX,XXX, in order to "guess" which tracks the user is most likely to request in the future. Alternative methods of determination may also be used. In some embodiments, the files the user will likely want may be

known, such as when the user listens to a personalized radio station, or tends to listen to a particular online radio station. Such radio stations often have a list of the files that will be streamed later. Thus, this list can be used to determine 2742 what files will be required.

[0039] After the jukebox 103 determines 2742 what files will likely be required, the jukebox 103 requests the second parts 2718 of the files at a time when the connection is idle, or when sufficient bandwidth exists to transmit files. The authorization and content server 1614 then transmits 2744 the requested second parts 2718 to the jukebox 103. The jukebox 103 locally stores 2746 the second parts 2718 to the files. At a later time, when a user requests that a program file 2710 be streamed, the jukebox 103 requests the first part 2716 of the program file 2710, as described above. In response, authorization and content server 1614 transmits 2748 the first part 2716. The jukebox 103 combines 2750 the received first part 2716 with the stored second part 2718 to form a representation of the file 2710 and plays back 2752 the representation of the file 2710. Such a technique permits higher-quality playback than would otherwise be available, by pre-transmitting a significant portion of the data needed for playing back the media program. As described above, in one embodiment, the recombined file 2710 and the first part 2716 are discarded 2754 after or during playback.

[0040] In one alternative embodiment, instead of transmitting 2744 the second parts 2718 over a network, the second parts 2718 may be stored on physical media (such as a compact disc or floppy disk). That physical media is then physically sent to the location of the jukebox 103, where it can act as a local storage device, or the second parts 2718 may be transferred from the physical media to desired local storage. Thus, even for slow network connections, or for very large files, the user may enjoy a streaming experience, while the server retains control of the program file.

[0041] In another embodiment, the file is further split according to circumstances surrounding its transmission. The stream delivery system 150 begins to transmit all or part of the program file, such as an audio file, to the jukebox 103. When the program file, or part of the program file is partially transmitted, transmission is stopped. This stop can occur because of a disconnection between the stream

delivery system 150 and the jukebox 103, because a user wishes to stop transmission and use the bandwidth for other purposes (such as when skipping to the next streamed song in a personalized radio station), or other reasons. The partially transmitted section of the program file or part of the program file is stored locally. Then, at a later time, the rest of the program file or part of the program file is transmitted and combined with the stored part. Such a process is more efficient than starting to transmit the entire program file or part of the program file from the beginning if it was not completely transmitted in a prior session.

[0042] In one embodiment, the size of the transmitted information is used to determine whether the entire file or section has been transmitted, and if not, what further information must be transmitted. At the beginning of transmission, the stream delivery system 150 sends information to the jukebox 103 indicating the size of the file and parts to be transmitted. This information may be sent to the jukebox 103 in other ways as well, as part of a sequence listing, for example. The jukebox 103 can then use the received size of the file in conjunction with the stored file size to determine what remaining information is needed from the stream delivery system 150. Other methods to identify what further information is needed to complete a partially received file may also be used.

[0043] For example, one embodiment splits a 1000 byte audio file into two parts, a first section 2716 of 50 bytes, and a second section 2718 of 950 bytes. The stream delivery system 150 transmits 800 bytes of the second section 2718 to the jukebox 103. At this point, a user chooses to start transmission of another audio file, and end transmission of the first audio file. The transmitted 800 bytes of the second section 2718 are stored locally. Thus, to allow the user to listen to the first audio file in its entirety, only the remaining 150 bytes of the second part 2718 need be transmitted to the jukebox. This subsequent transmission of the remaining 150 bytes may be done as needed, such as when a user begins to listen to the audio file again and the first 800 bytes are taken from local storage, then the last 150 bytes are received from the stream delivery system. This transmission can also be performed in advance of the time the user wishes to listen to the audio file, such as a scheme where the jukebox 103 periodically checks for partial file sections, such as partial second parts 2718, and auto-

matically request transmission of the rest of the second part 2718. The jukebox 103 may identify partial file sections by comparing the total file size to the stored size, and then requesting only the missing information, or through other methods.

[0044] In another alternative embodiment, instead of discarding the first part 2716, the jukebox 103 encrypts the first part 2716 and stores the first part 2716 locally. This further reduces the amount of data that must be transmitted for program file 2710 playback. The server 1614 may still retain control over playback of the program file 2710 by controlling when the jukebox 103 may decrypt the first part 2716, for recombination with the second part 2718 and playback of the program file 2710. The second part 2718 could also be encrypted in place of, or in addition to, the first part 2716.

[0045] In yet another embodiment, transmitter 121 employs scalable coding to increase the quality of audio output despite limitations in channel capacity. Fig. 4A is a block diagram illustrating an example of a scalably-coded program file, including data stored in two sections 2804, 2806 that together form a program file 2802. Primary set of data 2804 is sufficient to play back the program file 2802 at a low quality level. Secondary set of data 2806 may be added to the primary set 2804 for higher-quality playback. The sets of data 2804, 2806 may be stored separately in the content database 102 or with the signal files 1615, or may be stored in a combined state, to be separated and transmitted as needed.

[0046] Fig. 4B is a block diagram illustrating the use and combination of primary data 2804 and secondary data 2806 to provide different playback quality levels. As mentioned above, the primary data 2804 is sufficient to play back the program file 2802 at a low quality level 2808. In one embodiment, the secondary data 2806 alone does not provide information sufficient for playback 2810 of the program file 2802. However, when the secondary data 2806 is combined 2812 with the primary data 2804, the combined data 2812 is sufficient to play back the program file 2802 at a high quality level 2814. For example, if the program file 2802 is an audio file, low-quality audio can be produced using the primary data 2804. Audio cannot be produced with secondary information 2806 alone. If secondary information 2806 is combined with the primary information 2804, output quality is enhanced. In one em-

bodiment, additional levels of information may also be provided, each of which can be combined with the lower levels to further enhance output quality. This allows caching of lower-quality audio received in a first session and later receiving secondary information 2806 in a second session, which enables the jukebox 103 to combine the primary and secondary information to output audio of increased quality.

[0047] Since different sets of data are available that successively improve program file 2802 quality level, the quality of the output by the jukebox 103 depends on how much data can be transmitted in the time available. The first time an audio track is transmitted, transmitter 121 provides jukebox 103 with the primary information 2804 first, which allows the jukebox 103 to playback a representation of the program file 2802. Secondary 2806 (and additional) information is transmitted to allow improved quality levels as time permits. Jukebox 103 outputs the audio track with whatever level of information it has received at the time output is to commence. If only primary information 2804 has been received, jukebox 103 outputs lower-quality audio. If secondary information 2806 has also been received, it is combined with the primary information 2804 and jukebox 103 outputs higher-quality audio.

[0048] In addition, jukebox 103, in one embodiment, caches the received information. If the same audio track is requested at a later time, transmitter 121 provides jukebox 103 with the next level of information. Therefore, even if jukebox 103 was unable to provide higher-quality audio during the first listening, it may be able to provide higher-quality audio during subsequent listenings, by combining secondary (and/or additional) information with the previously cached primary information to generate the higher-quality audio output. Such a technique facilitates the output of high quality audio even when network transmission capacities are limited.

[0049] Referring now to Fig. 4C, there is shown an example of a transfer sequence for a channel with moderate bandwidth. Initially, tracks A and B are requested. Primary information for track A 2820 is downloaded. As primary information 2820 is downloaded, a low-quality representation of track A 2828 is played, according to conventional streaming audio techniques. Downloaded primary information 2820 is cached.

[0050] Once the download of primary information for track A 2820 is complete, jukebox 103 begins to download primary information for track B 2822. This download may begin even though track A is still playing 2828. In the example shown in Fig. 4C, the download of primary information for track B 2822 is completed while track A is still playing 2828. Therefore, jukebox 103 begins to download secondary information for track B 2824. Then, when playback 2828 of track A is finished, jukebox 103 is able to output a high quality representation of track B 2830, by combining secondary information 2824 with previously downloaded primary information 2822. The output of the high quality version 2830 may take place while secondary information 2824 is still being downloaded, again using streaming techniques.

[0051] In the example of Fig. 4C, a request to play track A a second time is received. Therefore, once secondary information 2824 has been downloaded, jukebox 103 begins to download secondary information for track A 2826. Once the high quality version of track B 2830 is finished playing, jukebox 103 outputs a high quality representation of track A 2832, by combining secondary information 2826 with previously downloaded primary information 2820.

[0052] Referring now to Fig. 4D, there is shown another example of a transfer sequence for a channel with a lower bandwidth than that of Fig. 4C. Here, the secondary information for track B 2824 is not downloaded, because it would not arrive in time to improve the output of track B. Accordingly, a lower quality version of track B 2834 is output in lieu of the higher quality version 2830 of Fig. 4C. However, the higher quality version of track A 2832 can still be presented, since there is sufficient time to download secondary information for track A 2826 before the second playback of track A commences.

[0053] One skilled in the art will recognize that the tracks depicted in Figs. 4C and 4D may refer to individual songs, or song segments, or any other unit of information. One skilled in the art will further recognize that the scalable coding techniques described herein may be applied to video data, or to any other type of data, and are not limited to audio data.

[0054] The scalable coding techniques of the present invention thus facilitate the trading off of quality in bandwidth-limited situations, without requiring

complex bandwidth estimation and determination. If insufficient bandwidth exists for the delivery of higher-quality versions, the system simply continues playing lower quality versions of tracks. No skipping, pausing, or other interruption of the audio stream is necessary. Jukebox 103 can determine whether to continue any particular transfer to improve the available quality or to download the next requested track, based on upcoming track selections. At any given moment, the next data segment to request can be determined by requesting the highest priority data segment from the next few audio segments. In one embodiment, priorities are defined to either play audio at a maximum short-term quality level or at a consistent quality level.

[0055] In one embodiment, jukebox 103 requests data for downloading according to the following order of priorities:

Priority	Type of value
1	Primary information, next track
2	Secondary information, next track
3	Primary information, track after next
4	Secondary information, track after next
5	Tertiary information, next track
6	Tertiary information, track after next
7	Data for subsequent tracks

[0056] One skilled in the art will recognize that any desired priority list may be provided. For example, if item 5 in this table is moved up to the third rank, the system will give more priority to high quality presentation at the possible expense of inconsistent quality on lower bandwidth connections.

[0057] In one embodiment, locally-cached downloaded data is stored in an encrypted or otherwise protected form, so as to prevent its abuse and to inhibit copyright infringement. In another embodiment, primary information is stored in an encrypted or otherwise protected form, but secondary and subsequent information is not, since the secondary and subsequent information is unusable without access to the primary information.

[0058] In one embodiment, jukebox 103 downloads data sets when the user is not actually listening to music, so as to facilitate improved usage of an otherwise idle network connection. Jukebox 103 determines which items are likely to be requested by a user, so that at idle times it can transfer data that is likely to be useful for rendering audio segments in the future. Such determination may be made in the same way as described above with respect to the split files of Fig. 3D. In one embodiment, secondary information 2806 for such "predicted" audio segments is downloaded first, so that encryption is not required unless and until the user actually requests the tracks and the primary information 2804 is to be downloaded. Scalable coding may also be used to process a signal of a conventional broadcast radio station that plays music. An audio recognition device, as is conventional, pre-processes the signal in order to identify individual songs. Those portions of audio information that are not music are compressed and stored, and a transfer sequence is sent to jukebox 103 that references these recently encoded non-music segments as well as previously known and cached musical segments. The recently encoded segments can be encoded at a lower quality level in order to allow a jukebox 103 connected by a low speed line to transfer the recently encoded segments in real-time while still playing the cached musical segments at a higher quality level.

[0059] As with file splitting, portions of scalably coded files may also be sent on physical media rather than transmitted over a network. Similarly, if sections of scalably coded information are only partially transmitted, they may be stored locally, and then only the information not previously received is transmitted in a subsequent session.

[0060] File splitting and scalable coding may also be used in combination. Fig. 5A is a block diagram illustrating how a program file 2902 may be divided into primary data 2904 and secondary data 2906, and the primary data 2904 file split into a first part 2908 and a second part 2910. The primary data 2904 is split as described above with respect to Figs. 3A and 3B. This allows almost all of the primary data 2904 and secondary data 2906 to be cached locally without requiring encryption. Since, in one embodiment, the second part 2910 of the primary data 2904 is not playable without the first part 2908, and the secondary data 2906 is not playable without a playable

set of primary data 2904, all of the program file 2902 except the first part 2908 of the primary data 2904 may be cached locally without providing the user with a usable copy of the program file 2902 that may be played back. This allows the server 1614 to retain control of playback of the program file 2902, as long as the first part 2908 of the primary data 2904 is not stored locally in a usable form.

[0061] Fig. 5B is a block diagram that illustrates how such a scheme provides advantages for streaming program files 2902. First, the first and second parts 2908, 2910 of the primary data 2904 are transmitted 2912 to the client by the server 1614. The first and second parts 2908, 2910 may be transmitted at roughly the same time in response to the jukebox 103 requesting the program file 2902. Alternatively, the second part 2910 may be transmitted in advance, after determination of what files 2902 are likely to be desired, as described above with respect to Fig. 3D. In an embodiment wherein the second part 2910 is transmitted in advance, the first part 2908 is transmitted in response to the jukebox 103 requesting the program file 2902. The first and second parts 2908, 2910 are sufficient to provide low quality playback 2914. The second part 2910 is then stored 2916 locally, and the first part 2908 is discarded. At a later time, such as when the jukebox 103 requests the program file 2902 again, the server 1614 transmits 2918 the first part of the primary data 2908 and the secondary data 2906. This allows sufficient information for the program file 2902 to be played back at a high quality level 2920. The secondary data 2906 is stored locally 2922, and the first part 2908 of the primary data 2904 is discarded. For subsequent playback of the program file 2902, only the first part 2908 of the primary data 2904 need be transmitted 2924 in order to enable high quality level playback 2920. Again, after such playback, the first part 2908 of the primary data 2904 is discarded, so that the server may retain control of program file 2902 playback. As an alternative to discarding the first part 2908, the first part 2908 may be encrypted, with decryption only occurring under the server's 1614 control.

[0062] Thus, a significant portion of the program file 2902 may be stored locally, with only the relatively small first part 2908 being transmitted from the server 1614 to provide the user with streaming file performance. This allows high quality streaming performance even over a relatively narrow bandwidth connection.

[0063] As described above, when scalable coding or file splitting is used, the jukebox 103 plays back a representation of the program file. This representation may be an exact digital replica of the program file, such as when the program file is split, all parts are transmitted to the jukebox 103, and the entire file is combined for playback. The representation may also be a less than exact replica, such as when only some portions of a scalably coded program file are combined and played back.

Recommendation Engine

[0064] One method of determining in advance what program files a user may want streamed is through use of a recommendation engine, described more fully in U.S. Patent Application Serial Number 09/846,823, filed on April 30, 2001, Patent No. X,XXX,XXX. Note that there are also many other ways to determine what program files a user may want streamed in advance. Referring now to Fig. 2, the jukebox 103 or personalized radio station accepts a user's selections of music tracks and makes additional recommendations as to music tracks the user is likely to enjoy. The user is able to search for particular tracks and/or artists, and to control the playback of selected tracks. The system monitors the user's behavior with regard to searching, listening, and playback control, and generates and analyzes logs of such behavior in order to refine recommendations. Advertising, offers, and other information may be selected and presented to the user based on observations of user behavior and analysis as to which material may be of interest to the user.

[0065] Compressed signal files 1615 contain descriptions of music tracks, and in one embodiment contains digitized representations of the music tracks themselves. Compressed signal files 1615 are stored, for example, using conventional database storage means or in a conventional file system, and in one embodiment include several fields providing descriptive information regarding music tracks, such as title, album, artist, type of music, track length, year, record label, and the like.

[0066] Selected tracks are played via jukebox 103, which is implemented in one embodiment as a standalone application, or as a plug-in or bundled feature in browser 105. Jukebox 103 receives digitized representations of music tracks and plays the tracks over a speaker or headphones at the user's computer. In one embodiment,

jukebox 103 can download and save music tracks in local storage (such as a hard disk drive, memory, or other local storage) in a compressed format, such as MP3, for playback on the user's computer or on a portable digital music listening device.

[0067] Recommendation engine 107 generates track preferences based on user information. Radio sequence generator 1613 uses track preferences, along with general constraints 1616 and format definitions 1611, to generate a sequence of tracks to be played. General constraints 1616 include particular rules and restrictions on the sequence of tracks, as may be required by law or as may be determined to be desirable for marketing or aesthetic purposes or for other reasons. Examples of constraints 1616 include: "no more than one song per hour from a particular album," or "do not play a fast song immediately after a slow song." Radio sequence generator 1613 may also incorporate a randomization element, if desired, and may be configurable by a website operator.

[0068] The track list is sent to jukebox 103 to be played to the user. A user activates jukebox 103 and selects music tracks for playback and/or purchase, via a user interface including controls and selectors. Authorization and content server 1614 checks that the appropriate security measures are in place (in order to prevent the user from "hacking" jukebox 103 to request unauthorized tracks from content server 1614), obtains the actual music tracks from files 1615, and provides them to jukebox 103 for output.

[0069] In one embodiment, the connections among the various elements of Figs. 1A and 2 are implemented over the Internet, using known protocols such as HTTP and TCP/IP. Secure sockets layer (SSL) or other encryption techniques may be employed for added security.

[0070] The recommendation engine may be employed in connection with conventional radio station programming techniques, to implement an improved personalized radio station. As is known in the art, conventional radio stations typically divide a programming block into a number of segments. Each segment is assigned a programming category, such as "power hit," "new release," "recurrent hit," and the like. For a particular programming block, music tracks are assigned to each of the segments based on the particular programming format of the radio station. Music

scheduling software, such as Selector® by RCS Sound Software, applies heuristic rules for repetition limits and classes of songs, to automatically generate track lists for use by radio stations. The recommendation engine may be combined with such existing radio station programming techniques, to populate the defined segments with music tracks that are likely to appeal to a particular listener. Additional rules may be applied in generating track lists, so as to limit undesired repetition and to comply with limiting legislation (such as the Digital Millennium Copyright Act) and other restrictions.

[0071] From the above description, it will be apparent that the invention disclosed herein provides a novel and advantageous system and method for providing a user with a streaming program file experience. The foregoing discussion discloses and describes merely exemplary methods and embodiments of the present invention. As will be understood by those familiar with the art, the invention may be embodied in other specific forms without departing from the spirit or essential characteristics thereof. For example, the invention may be applied to other domains and environments, and may be employed in connection with additional applications where personalized recommendations are desirable. Accordingly, the disclosure of the present invention is intended to be illustrative, but not limiting, of the scope of the invention, which is set forth in the following claims.

[0072] What is claimed is:

1. In a client connected to a network and including a local storage device, a computer-implemented method of receiving and playing a media file comprising at least a first part and a second part, comprising:

receiving over the network and from a remote server in a first session the second part of the media file;

storing the second part at the local storage device;

receiving over the network and from a remote server in a second session subsequent to the first session the first part of the media file and not the second part of the media file;

combining the first part received in the second session with the stored second part after the start of the second session; and

playing a representation of the media file at the local client after combining at least part of the first and second parts after the start of the second session.

2. The computer-implemented method of claim 1, wherein the second part is not sufficient to properly play the representation of the media file back without being combined with the first part.

3. The computer-implemented method of claim 2, wherein the received parts of the media file are encoded, and the second part may not be decoded without being combined with the first part.

4. The computer-implemented method of claim 1, wherein the representation of the media file may not be played back at the local client without authorization by a remote server.

5. The computer-implemented method of claim 4, wherein authorization comprises the reception over the network and from a remote server the first part of the media file.

6. The computer-implemented method of claim 1, wherein the neither the first part nor the second part may be used independently to recreate a recognizable facsimile of the content of the media file.

7. The computer-implemented method of claim 1, wherein the media file is divided into a plurality of sections, each section comprising a plurality of bits, wherein the first part comprises a first subset of bits from each section, and the second part comprises the remaining bits from each section.

8. The computer-implemented method of claim 1, wherein the media file is divided into a plurality of sections, each section comprising a plurality of bits, wherein the first part comprises a first bit from each section, and the second part comprises the remaining bits from each section.

9. The computer-implemented method of claim 7 or 8, wherein one of the sections contains exactly X bits, and wherein each of the remaining sections contains exactly Y bits, wherein X is less than or equal to Y.

10. The computer implemented method of claim 9, wherein Y equals 20.

11. The computer implemented method of claim 1, wherein the first part comprises headers, compression table selectors and scale factors of the media file.

12. The computer-implemented method of claim 1, further comprising:
receiving over the network and from a remote server during the first session the first part of the media file; and
playing the representation of the media file back at the local client during the first session.

13. The computer-implemented method of claim 1 or 12, wherein the first part is not stored at the local storage device after the playing of the media file.

14. The computer-implemented method of claim 1 or 12, wherein the combined first and second parts are not stored at the local storage device after the playing of the media file.

15. The computer-implemented method of claim 1, wherein the played representation of the media file is an exact digital copy of the media file.

16. The computer implemented method of claim 1, wherein:

the media file further comprises a third part;

the first and second parts are combinable to form a representation of the media file at a first quality level; and

the first, second, and third parts are combinable to form a representation of the media file at a second quality level higher than the first quality level.

17. The computer implemented method of claim 16, further comprising:

receiving over the network and from a remote server the third part of the media file;

storing the third part at the local storage device;

combining the received third part with the stored first part and the stored second part to form the representation of the media file at the second quality level; and

playing the representation of the media file after combining the first, second, and third parts.

18. The computer implemented method of claim 17, wherein the third part is received over the network and from a remote server in a session other than the first session.

19. The computer implemented method of claim 17, wherein the third part and not the second part is received over the network and from a remote server in a session other than the first session.