US 20030221127A1

(19) United States

(12) Patent Application Publication (10) Pub. No.: US 2003/0221127 A1

Risan et al. (43) Pub. Date: Nov. 27, 2003

(54) SYSTEM AND METHOD FOR PROVIDING GLOBAL MEDIA CONTENT DELIVERY

(76) Inventors: **Hank Risan**, Santa Cruz, CA (US); **Edward Vincent Fitzgerald**, Santa Cruz, CA (US)

Correspondence Address:
WAGNER, MURABITO & HAO LLP
Third Floor
Two North Market Street
San Jose, CA 95113 (US)

(21) Appl. No.: **10/235,293**

(22) Filed: **Sep. 4, 2002**

Related U.S. Application Data

(60) Provisional application No. 60/323,468, filed on Sep. 18, 2001.

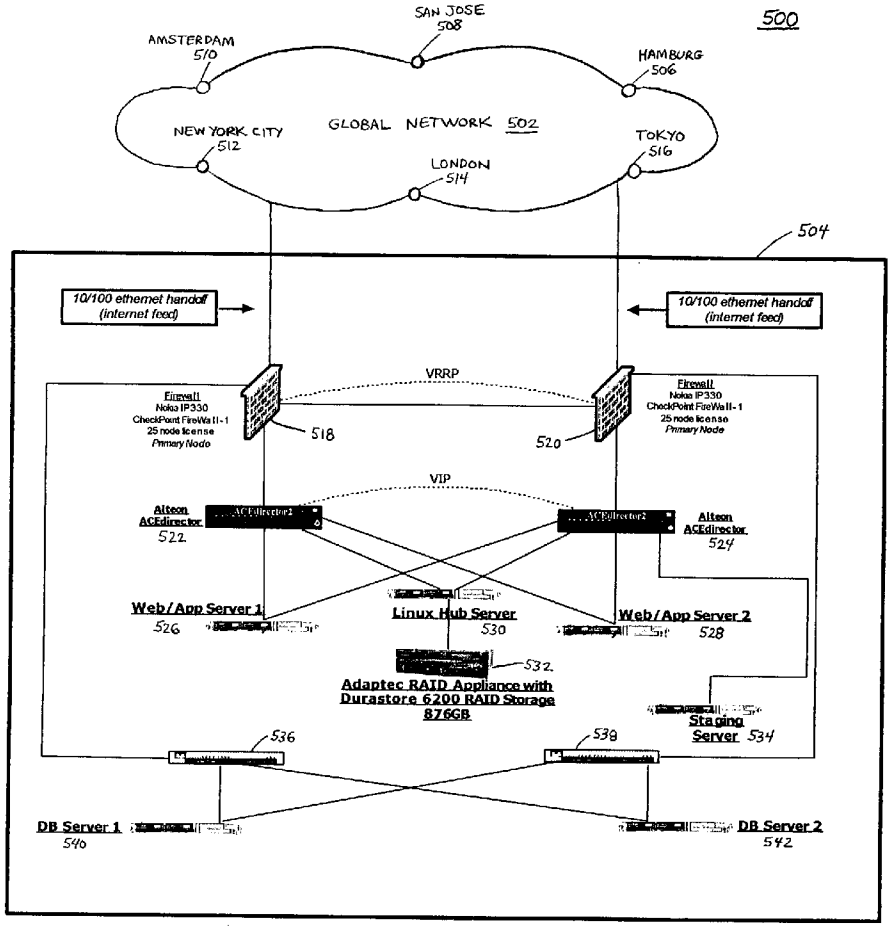
Publication Classification

(51) **Int. Cl.**[7] **H04L 9/00**

(52) **U.S. Cl.** **713/201**

(57) **ABSTRACT**

One embodiment of the present invention enables global delivery of "on-demand" high fidelity media content to client computers via a network, such as, the Internet or a wide area network (WAN) while restricting unauthorized users from directly retrieving media content from its sources. The present embodiment includes a global media content delivery network that may include multiple "points of presence" which may be located throughout the world. Each point of presence may store a portion or the entirety of a media content library that may be provided to client devices. Each one of the points of presence may provide media content to client devices in their respective vicinity of the world. Once a client receives media, it is stored using hidden directories to prevent easy redistribution with other devices. An access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media requests.



GOOGLE EXHIBIT 1008

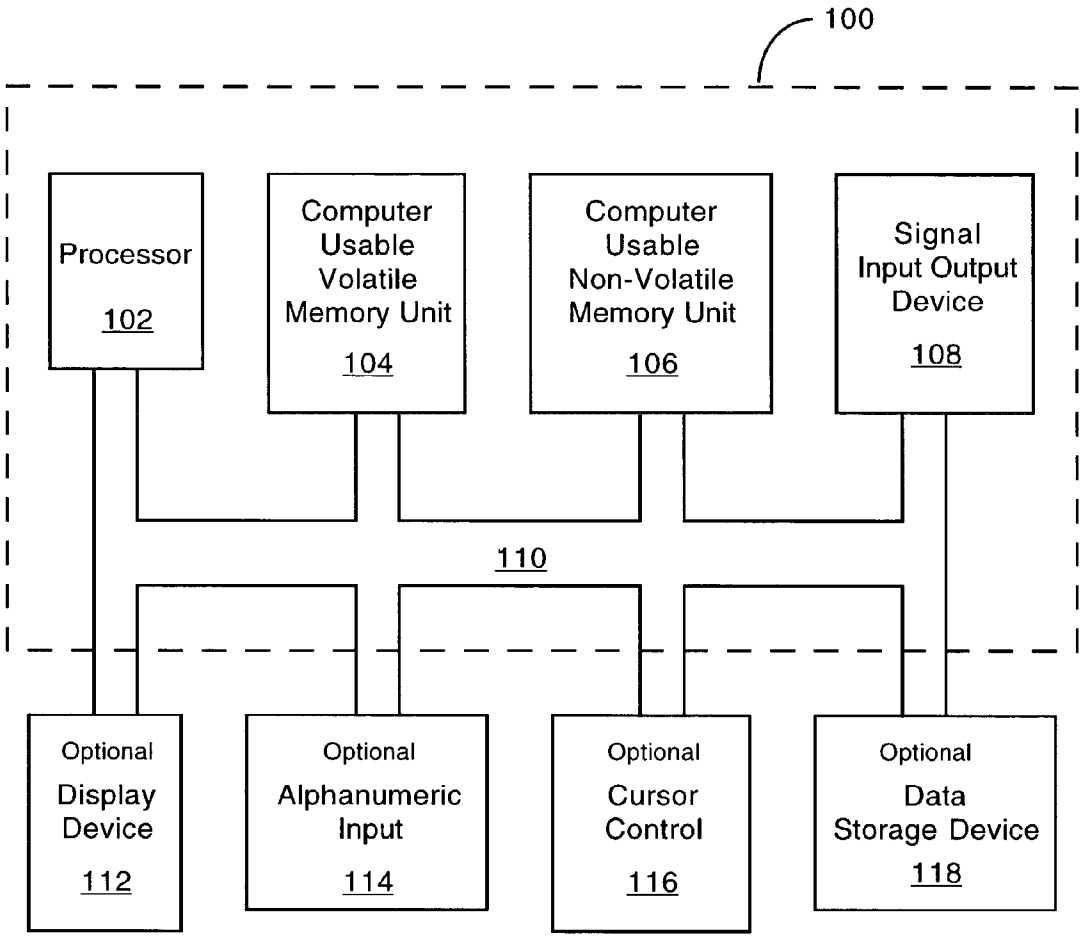

100
Processor
102
Computer Usable Volatile Memory Unit
104
Computer Usable Non-Volatile Memory Unit
106
Signal Input Output Device
108
110
Optional Display Device
112
Optional Alphanumeric Input
114
Optional Cursor Control
116
Optional Data Storage Device
118

Fig. 1

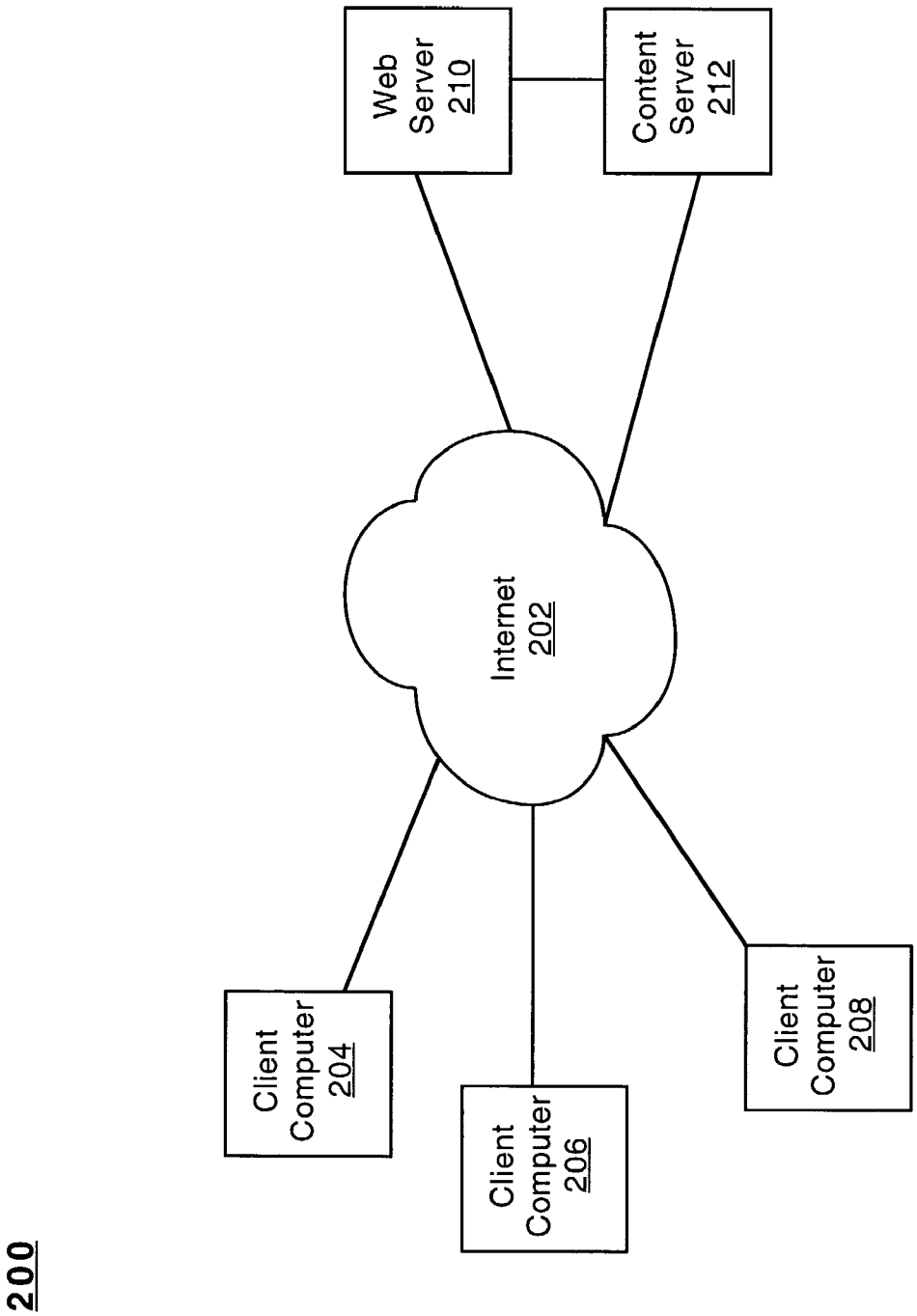

200
Client Computer 204
Client Computer 206
Client Computer 208
Internet 202
Web Server 210
Content Server 212

Fig. 2

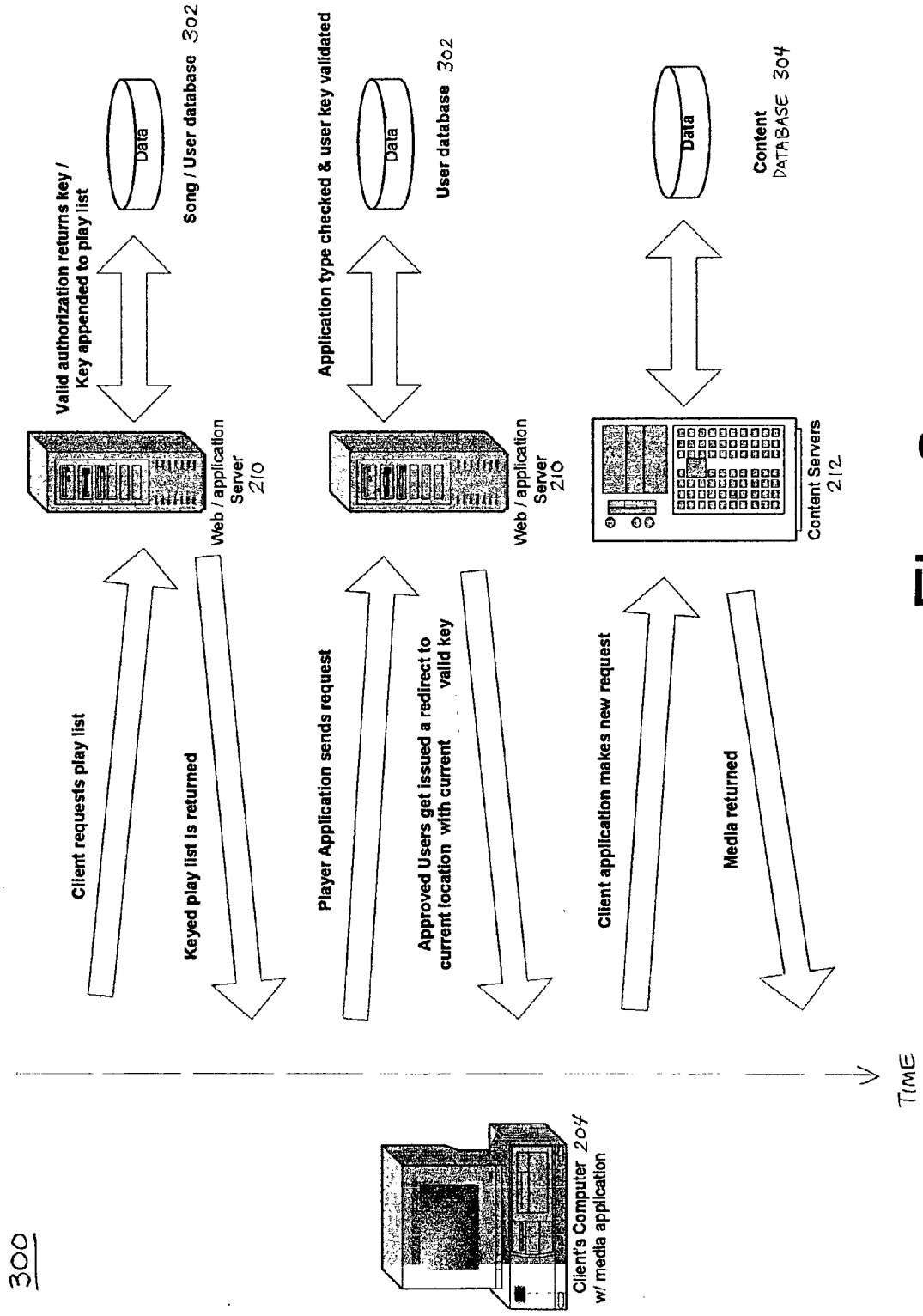

300
Client's Computer 204
w/ media application
Client requests play list
Keyed play list is returned
Web / application Server 210
Valid authorization returns key / Key appended to play list
Data
Song / User database 302
Player Application sends request
Approved Users get issued a redirect to current location with current valid key
Web / application Server 210
Application type checked & user key validated
Data
User database 302
Client application makes new request
Media returned
Content Servers 212
Data
Content DATABASE 304
TIME

Fig. 3

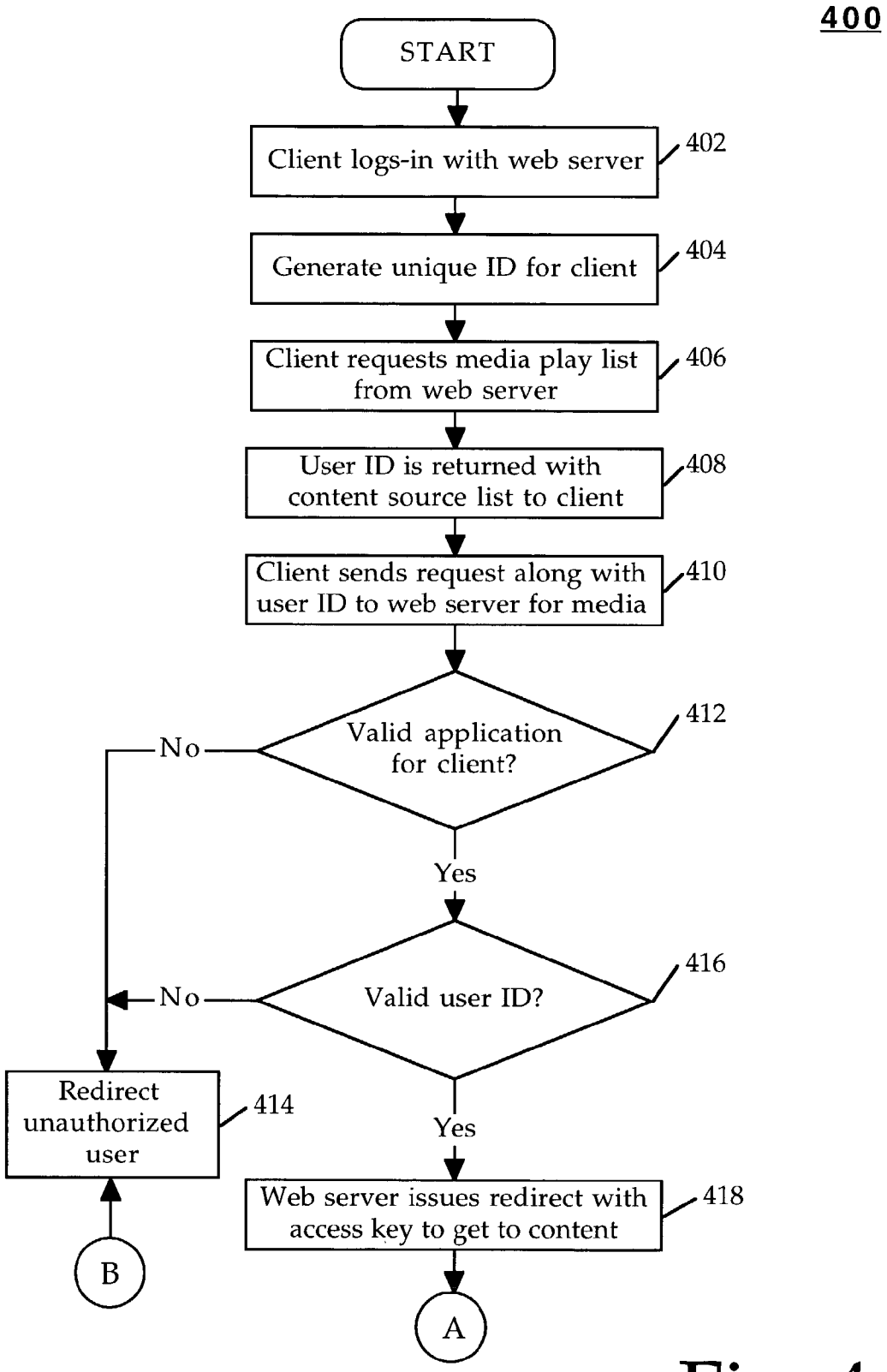

400
START
Client logs-in with web server
402
Generate unique ID for client
404
Client requests media play list from web server
406
User ID is returned with content source list to client
408
Client sends request along with user ID to web server for media
410
Valid application for client?
412
No
Yes
Valid user ID?
416
No
Yes
Redirect unauthorized user
414
Web server issues redirect with access key to get to content
418
B
A

Fig. 4A

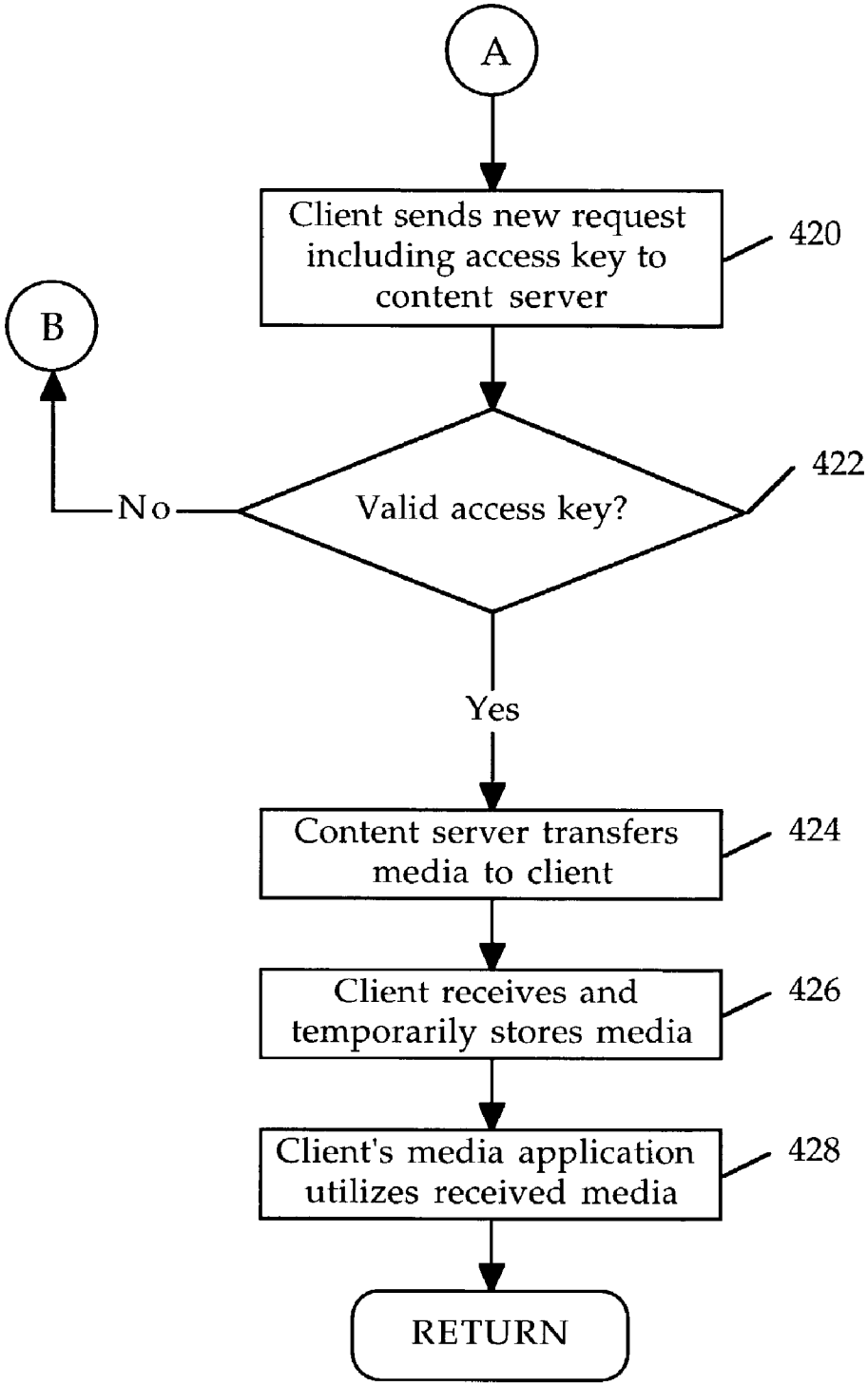

A
Client sends new request including access key to content server
420
B
No
Valid access key?
422
Yes
Content server transfers media to client
424
Client receives and temporarily stores media
426
Client's media application utilizes received media
428
RETURN

Fig. 4B

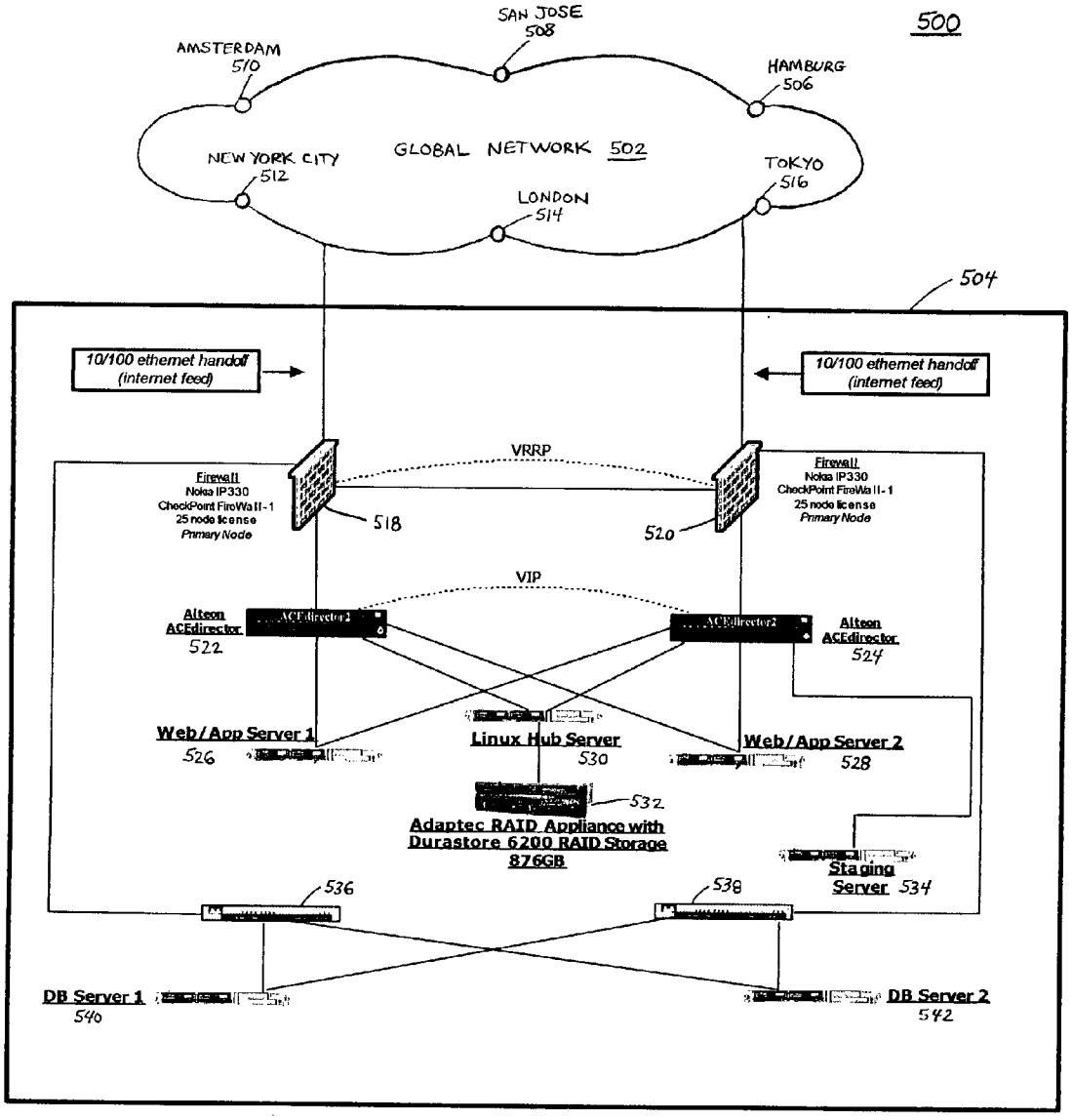

500
SAN JOSE
508
AMSTERDAM
510
HAMBURG
506
GLOBAL NETWORK 502
NEW YORK CITY
512
TOKYO
516
LONDON
514
504
10/100 ethernet handoff (internet feed)
10/100 ethernet handoff (internet feed)
VRRP
Firewall
Nokia IP330
CheckPoint FireWall-1
25 node license
Primary Node
518
520
Firewall
Nokia IP330
CheckPoint FireWall-1
25 node license
Primary Node
VIP
Alteon ACEdirector
522
Alteon ACEdirector
524
Web/App Server 1
526
Linux Hub Server
530
Web/App Server 2
528
532
Adaptec RAID Appliance with Durastore 6200 RAID Storage 876GB
Staging Server 534
536
538
DB Server 1
540
DB Server 2
542

Fig. 5

SYSTEM AND METHOD FOR PROVIDING GLOBAL MEDIA CONTENT DELIVERY

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application claims priority to the copending U.S. Provisional Patent Application No.: 60/323,468 entitled "User Selectable and Location-Obscured Streaming Media Method and System" by Hank Risan et al., filed Sep. 18, 2001, Attorney Docket Number: MOMI-001.PRO; copending U.S. Provisional Patent Application No.:_____ entitled "Method and System for Providing User Selectable and Location-Obscured Streaming Media" by Hank Risan et al., filed May 10, 2002, Attorney Docket Number: MOMI-002.PRO; and copending U.S. Provisional Patent Application No.:_____ entitled "System and Method for Providing User Selectable and Location-Obscured Streaming Media" by Hank Risan et al., filed May 10, 2002, Attorney Docket Number: MOMI-003.PRO, which are hereby incorporated by reference.

TECHNICAL FIELD

[0002] The present invention generally relates to the field of media. More particularly, the present invention relates to the field of media delivery.

BACKGROUND ART

[0003] Computers have become integral tools used in a wide variety of different applications, such as in finance and commercial transactions, computer-aided design and manufacturing, health care, telecommunication, education, etc. Computers are finding new applications as a result of advances in hardware technology and rapid development in software technology. Furthermore, the functionality of a computer system is dramatically enhanced by coupling these type of stand-alone devices together in order to form a networking environment. Within a networking environment, users may readily exchange files, share information stored on a common database, pool resources, and communicate via electronic mail (e-mail) and video teleconferencing. Furthermore, computers which are coupled to a networking environment like the Internet provide their users access to data and information from all over the world. Computer systems have become useful in many aspects of everyday life including providing entertainment to their users.

[0004] For example, a computer system may be coupled to the Internet and receive music from a wide variety of web site sources. There are different ways that a computer system may receive music via the Internet. One common way is for the computer to receive streaming music from a web site wherein the computer user typically does not choose the songs that he or she listens to but instead receives broadcast music. However, it should be appreciated that there are disadvantages associated with this streaming media technique for receiving music.

[0005] One of the disadvantages is that the computer user is unable to choose specific songs to listen to and have them delivered to his or her computer "on-demand." Another disadvantage with the streaming music delivery technique is that as more and more computers receive the simultaneously broadcast music, the more and more congested the communication networks of the Internet become. In order to lessen the severity of this type of network congestion, media streamers usually decrease the data rate of the streaming audio in order to be able to handle more clients. However, by decreasing the data rate of the music, its fidelity is also degraded.

[0006] Some streaming music delivery systems require a participating computer system to acquire a proprietary audio player in order to receive and play music which has been encrypted so that the music is not distributed to others in an uncontrolled fashion. Nevertheless, there are disadvantages associated with this technique also. For example, one of the disadvantages is that the fidelity of the received music can be degraded as a consequence of its encryption and decryption. Additionally, an encryption process typically involves more data being included with the streaming music which may provide additional congestion to communication networks of the Internet. Another disadvantage associated with some proprietary audio players is that they are not compatible with certain computer operating system platforms which consequently restricts the number of computer systems that may utilize them.

[0007] Another technique for acquiring music with a computer is by accessing a music file sharing web site via the Internet. Currently there are a variety of different Internet music file sharing web sites accessible to computers. Typically, these file sharing web sites enable computer users to share music files in a point-to-point manner thereby enabling them to receive music which may then be stored by their computer for use or further distribution. It should be appreciated that there are disadvantages associated with this type of music file sharing technique.

[0008] One of the disadvantages is that the music file sharing computer users usually do not compensate the owner (e.g., record company, artist, etc.) of copyrighted music after receiving a copy of it. One of the reasons for this situation is that music file sharing web sites commonly do not control or even monitor the music being exchanged among different computers. And even when these web sites try to record and/or control the distribution of music, it may be thwarted by simply changing the file name of the song thereby enabling it to continue to be distributed freely.

[0009] Another disadvantage associated with the music file sharing technique previously described is that the music received may be a low fidelity product. There are different causes for this low fidelity. For example, there is usually no established quality level that has to be met in order to share music files via the Internet. As such, the original "source" of a shared music file is typically music copied from a compact disc (CD) to a computer which may be done in a substandard manner because an operator is not concerned with quality and/or does not even have the tools necessary to produce a high fidelity product. A further disadvantage associated with this music file sharing technique is that as more and more file sharing or swapping occurs, communication network systems of the Internet can get increasingly more congested thereby reducing the efficiency of data transfer between different computers.

SUMMARY OF THE INVENTION

[0010] Accordingly, a need exists for a method and system for delivering "on-demand" high fidelity music to computer systems via the Internet which does not involve proprietary

audio players and/or encryption of the music. A further need exists for a method and system which meets the above need and does not overly congest the communication networks of the Internet. Another need exists for a method and system which meets the above needs and monitors the music delivered in order to compensate the owner of copyrighted music for it. The present invention provides one or more embodiments which accomplish the above mentioned needs.

[0011] Specifically, one embodiment of the present invention enables global delivery of "on-demand" high fidelity media content to client computers via a network, such as, the Internet or a wide area network (WAN) while restricting unauthorized users from directly retrieving media content from its sources. The present embodiment includes a global media content delivery network that may include multiple "points of presence" which may be located throughout the world. Each point of presence may store a portion or the entirety of a media content library that may be provided to client devices. Each one of the points of presence may provide media content to client devices in their respective vicinity of the world. Once a client receives media, it is stored using hidden directories to prevent easy redistribution with other devices. An access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media requests. By implementing these and other functionality, the present embodiment restricts redistribution of delivered media content and provides a means for compensating owners of copyrighted media content.

[0012] In another embodiment, the present invention provides a system for providing media content while preventing unauthorized access to a plurality of content sources. The system includes a media content library having a plurality of media content pieces. Furthermore, the system includes a global media content delivery network having the plurality of content sources. It is understood that each content source of the plurality of content sources stores a portion of the media content library. Additionally, the system includes a security system coupled to the plurality of content sources for allowing delivery of a media content piece from the media content library to an authorized client device while preventing unauthorized access to the media content library stored by the plurality of content sources.

[0013] In yet another embodiment, the present invention provides a method for providing media content to a plurality of client devices from a plurality of content sources while preventing unauthorized access to the media content. The method includes transmitting a first request for a media content list of a media content library from a client device of the plurality of client devices. In response to receiving the first request, the method includes transmitting to the client device the media content play list together with a unique identification. Furthermore, in response to receiving the media content list, the method includes transmitting a second request for delivery of the media content together with the unique identification from the client device. Provided the unique identification is valid, the method includes transmitting to the client device an access key together with a location of a content source of the plurality of content sources containing the media content. Additionally, in response to receiving the access key together with the location of the content source, the method includes transmitting a third request together with the access key to the content source from the client device. Provided the access key is valid, the method includes transmitting the media content to the client device.

[0014] In still another embodiment, the present invention provides a system for providing media content while restricting unauthorized access to a plurality of content sources. The system includes a media content library having a plurality of media content pieces. Additionally, the system includes a global media content delivery network having the plurality of content sources. It is appreciated that each content source of the plurality of content sources stores a portion of the media content library. Furthermore, the system includes an authorization system coupled to the plurality of content sources for regulating delivery of a media content piece from the media content library to an authorized client computer while restricting unauthorized access to the media content library stored by the plurality of content sources.

[0015] While particular embodiments of the present invention have been specifically described within this summary, it is noted that the invention is not limited to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents which may be included within the spirit and scope of the invention as herein described by the specification and Claims.

BRIEF DESCRIPTION OF THE DRAWINGS

[0016] The accompanying drawings, which are incorporated in and form a part of this specification, illustrate embodiments of the invention and, together with the description, serve to explain the principles of the invention.

[0017] **FIG. 1** is a block diagram of an embodiment of an exemplary computer system that may be used in accordance with the present invention.

[0018] **FIG. 2** is a block diagram of an exemplary network that may be used in accordance with an embodiment of the present invention.

[0019] **FIG. 3** is a block diagram of an exemplary system for implementing access keys in accordance with an embodiment of the present invention.

[0020] **FIGS. 4A and 4B** are a flowchart of steps performed in accordance with an embodiment of the present invention for providing media content to a client computing device.

[0021] **FIG. 5** is a diagram of an exemplary global media content delivery system in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0022] Reference will now be made in detail to embodiments of the invention, examples of which are illustrated in the accompanying drawings. While the invention will be described in conjunction with embodiments, it will be understood that they are not intended to limit the invention to these embodiments. On the contrary, the invention is intended to cover alternatives, modifications and equivalents, which may be included within the spirit and scope of the invention as defined by the appended claims. Furthermore, in the following detailed description of the present invention, numerous specific details are set forth in order to

provide a thorough understanding of the present invention. However, it will be obvious to one of ordinary skill in the art that the present invention may be practiced without these specific details. In other instances, well known methods, procedures, components, and circuits have not been described in detail as not to unnecessarily obscure aspects of the present invention.

[0023] Some portions of the detailed descriptions which follow are presented in terms of procedures, logic blocks, processing, and other symbolic representations of operations on data bits within a computing system or digital system memory. These descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art. A procedure, logic block, process, etc., is herein, and generally, conceived to be a self-consistent sequence of steps or instructions leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these physical manipulations take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated in a computing system or similar electronic computing device. For reasons of convenience, and with reference to common usage, these signals are referred to as bits, values, elements, symbols, characters, terms, numbers, or the like with reference to the present invention.

[0024] It should be borne in mind, however, that all of these terms are to be interpreted as referencing physical manipulations and quantities and are merely convenient labels and are to be interpreted further in view of terms commonly used in the art. Unless specifically stated otherwise as apparent from the following discussions, it is understood that throughout discussions of the present invention, discussions utilizing terms such as "allowing" or "preventing" or "restricting" or "utilizing" or "regulating" or "providing" or "determining" or "controlling" or "transmitting" or "receiving" or "recognizing" or "generating" or "storing" or the like, refer to the action and processes of a computing system, or similar electronic computing device, that manipulates and transforms data. The data is represented as physical (electronic) quantities within the computing system's registers and memories and is transformed into other data similarly represented as physical quantities within the computing system memories or registers or other such information storage, transmission, or display devices.

[0025] **FIG. 1** is a block diagram of an embodiment of an exemplary computer system **100** that may be used in accordance with the present invention. It is understood that system **100** is not strictly limited to be a computer system. As such, system **100** of the present embodiment is well suited to be any type of computing device (e.g., server computer, desktop computer, laptop computer, portable computing device, etc.). Within the discussions of the present invention, certain processes and steps are discussed that are realized, in one embodiment, as a series of instructions (e.g., software program) that reside within computer readable memory units of computer system **100** and executed by a processor(s) of system **100**. When executed, the instructions cause computer **100** to perform specific actions and exhibit specific behavior which is described in detail herein.

[0026] Computer system **100** of **FIG. 1** comprises an address/data bus **110** for communicating information, one or more central processors **102** coupled with bus **110** for processing information and instructions. Central processor unit(s) **102** may be a microprocessor or any other type of processor. The computer **100** also includes data storage features such as a computer usable volatile memory unit **104**, e.g., random access memory (RAM), static RAM, dynamic RAM, etc., coupled with bus **110** for storing information and instructions for central processor(s) **102**, a computer usable non-volatile memory unit **106**, e.g., read only memory (ROM), programmable ROM, flash memory, erasable programmable read only memory (EPROM), electrically erasable programmable read only memory (EEPROM), etc., coupled with bus **110** for storing static information and instructions for processor(s) **102**.

[0027] System **100** also includes one or more signal generating and receiving devices **108** coupled with bus **110** for enabling system **100** to interface with other electronic devices. The communication interface(s) **108** of the present embodiment may include wired and/or wireless communication technology. For example, in one embodiment of the present invention, the communication interface **108** is a serial communication port, but could also alternatively be any of a number of well known communication standards and protocols, e.g., a Universal Serial Bus (USB), an Ethernet adapter, a FireWire (IEEE 1394) interface, a parallel port, a small computer system interface (SCSI) bus interface, an infrared (IR) communication port, a Bluetooth wireless communication adapter, a broadband connection, and the like. In another embodiment a digital subscriber line (DSL) connection may be employed. In such a case the communication interface(s) **108** may include a DSL modem. In any case, the communication interface(s) **108** may provide a communication interface to the Internet.

[0028] Optionally, computer system **100** can include an alphanumeric input device **114** including alphanumeric and function keys coupled to the bus **110** for communicating information and command selections to the central processor(s) **102**. The computer **100** can also include an optional cursor control or cursor directing device **116** coupled to the bus **110** for communicating user input information and command selections to the central processor(s) **102**. The cursor directing device **116** can be implemented using a number of well known devices such as a mouse, a track ball, a track pad, an optical tracking device, a touch screen, etc. Alternatively, it is appreciated that a cursor can be directed and/or activated via input from the alphanumeric input device **114** using special keys and key sequence commands. The present embodiment is also well suited to directing a cursor by other means such as, for example, voice commands.

[0029] The system **100** of **FIG. 1** can also include a computer usable mass data storage device **118** such as a magnetic or optical disk and disk drive (e.g., hard drive or floppy diskette) coupled with bus **110** for storing information and instructions. An optional display device **112** is coupled to bus **110** of system **100** for displaying video and/or graphics. It should be appreciated that optional display device **112** may be a cathode ray tube (CRT), flat panel liquid crystal display (LCD), field emission display (FED), plasma display or any other display device suitable for

displaying video and/or graphic images and alphanumeric characters recognizable to a user.

[0030] **FIG. 2** is a block diagram of an exemplary network **200** that may be used in accordance with an embodiment of the present invention. For example, network **200** of the present embodiment enables one or more client computer systems (e.g., **204**, **206** and/or **208**) to receive high fidelity media content "on-demand" from a media content server **212** via the Internet **202** while restricting unauthorized users from directly retrieving media content from its source database stored by content server **212**.

[0031] Network **200** of **FIG. 2** includes web application server **210** and content server **212** which are communicatively coupled to the Internet **202**. Additionally, web application server **210** and content server **212** may be communicatively coupled together within out utilizing Internet **202**. Furthermore, network **200** includes client computers **204**, **206** and **208** which are each communicatively coupled to the Internet **202**. In this manner, web server **210**, content server **212**, and client computers **204-208** are able to communicate. It should be appreciated that the devices of network **200** of the present embodiment are well suited to be coupled in a wide variety of implementations. For example, content server **212**, web server **210**, and client computers **204-208** of network **200** may be coupled via wired communication technology (e.g., coaxial cable, copper wire, fiber optics, and the like) and/or wireless communication technology.

[0032] Within network **200**, it is understood that web server **210**, content server **212** and client computers **204-208** may be implemented in a variety ways in accordance with the present embodiment. For example, web server **210**, content server **212** and client computers **204-208** may be implemented in a manner similar to computer system **100** of **FIG. 1**. However, these devices of network **200** are not strictly limited to such implementation. Additionally, web server **210** and content server **212** of the present embodiment perform a variety of functionality within network **200**. It is understood that web server **210** and content server **212** may actually reside on a single physical computing device (e.g., computer **100** of **FIG. 1**). However, web server **210** and content server **212** of the present embodiment may each be implemented as one or more physical computing devices.

[0033] It is appreciated that network **200** of **FIG. 2** is able to operate with and provide delivery of any type of media content (e.g., audio, video, multimedia, graphics, information, data, software programs, and/or the like) in any type of format. For example, the content server **212** may provide audio clips, video clips, and the like to client computers **204-208** via the Internet **202**.

[0034] **FIG. 3** is a block diagram of an exemplary system **300** for implementing access keys in accordance with an embodiment of the present invention. Specifically, system **300** illustrates a client computer system (e.g., **204**) utilizing access keys in order to receive high fidelity media content "on-demand" from a media content server (e.g., **212**) while unauthorized users or media applications are restricted from directly retrieving and/or copying media content from its media source database (e.g., **304**).

[0035] The client computer device **204** may communicatively couple with a web/application server (e.g., **210**) in order to request a media play list. In response, the web server **210** may determine whether the user of client **204** is authorized to receive the media play list associated with the request. The web server **210** may utilize a user database **302** in order to determine if a user is authorized to receive a media play list. If not, web server **210** does not provide client computer **204** the requested media play list. However, if client **204** is authorized, web server **210** transmits the media play list to the client device **204** (e.g., via Internet **202**) along with an appended user key or user identification (ID). It is understood that user database **302** may also include data for one or more media play lists that may be utilized to provide a media play list to client computer **204**. Subsequently, the user of client device **204** may utilize the received media play list in combination with a media player application operating on client device **204** in order to transmit a delivery request (that automatically includes the user key or user ID) for one or more desirable pieces of media content to web server **210**.

[0036] Upon reception of the media content delivery request, the web server **210** of **FIG. 3** checks the validity of the requesting media application along with the attached user key. The web server **210** may utilize user database **302** in order to check the validity of the user key (or user ID) and the requesting media application. If either or both are invalid, web server **210** redirects the unauthorized client computer **204** so that it may be handled in an appropriate manner in order to prevent abuse of the system. However, if both the requesting media application and user ID are valid, web server **210** issues to client device **204** a redirect command to the current address location of the desired media content along with a time sensitive access key (e.g., for that day, hour, or for any defined timeframe).

[0037] In response to reception of the redirect command, the media player application operating on client computer device **204** automatically issues a new request along with the time sensitive access key to content server **212** for delivery of the one or more desired pieces of media content. The content server **212** determines the validity of the time sensitive access key. If invalid, content server **212** redirects the unauthorized client device **204** so that it may be handled in an appropriate manner in order to prevent access to the media content. However, if the time sensitive access key is valid, content server **212** retrieves the desired media content from a media content database **304** and then transmits it to client device **204** for use by its media player application. It is understood that the delivered media content may be stored by client computer **204** using hidden directories thereby preventing future unauthorized distribution of it.

[0038] **FIGS. 4A and 4B** are a flowchart **400** of steps performed in accordance with an embodiment of the present invention for providing media content to a client computing device (e.g., **204**, **206** or **208**). Flowchart **400** includes processes of the present invention which, in one embodiment, are carried out by a processor(s) and electrical components under the control of computer readable and computer executable instructions. The computer readable and computer executable instructions may reside, for example, in data storage features such as computer usable volatile memory **104**, computer usable non-volatile memory **106** and/or computer usable mass data storage device **118**. However, the computer readable and computer executable instructions may reside in any type of computer readable medium. Although specific steps are disclosed in flowchart

400, such steps are exemplary. That is, the present invention is well suited to performing various other steps or variations of the steps recited in **FIGS. 4A and 4B**. Within the present embodiment, it should be appreciated that the steps of flowchart **400** may be performed by software, by hardware or by any combination of software and hardware.

[**0039**] The present embodiment enables delivery of "on-demand" high fidelity media content to computer devices via one or more communication networks (e.g., the Internet) while restricting unauthorized users from directly retrieving media content from a source database. Once the computer device receives the media, it may be stored using hidden directories so that it may not be easily shared with others. Within the present embodiment, there are different functionality that are implemented in order to protect and monitor the media content source. For example, the actual address location of the media database is hidden from content recipients while its address directory is periodically change making past addresses obsolete. Furthermore, an access key procedure and rate control restrictor may also be implemented to monitor and restrict suspicious media content requests. By implementing these and other functionality, the present embodiment restricts access to and redistribution of delivered media content and provides a means for compensating owners of copyrighted media content.

[**0040**] It should be appreciated that flowchart **400** is described in conjunction with **FIG. 2** in order to more fully describe the operation of the present embodiment. At step **402** of **FIG. 4A, a** client computer (e.g., **204**) communicatively couples to a web server (e.g., **210**) via one or more communication networks (e.g., Internet **202**) and then proceeds to log in. The initial log in process of step **402** may be implemented in a wide variety of ways in accordance with the present embodiment. For example, the initial log in process may involve a registration process provided by web server **210** wherein different identifying information may be provided by a user of client computer **204** such as, but not limited to, his or her name, address, phone number and credit card information. The present embodiment may then cause web server **210** to verify the truthfulness of the provided registration information. Furthermore, the registration process may include the user of client device **204** establishing with web server **210** a user selected identity and password.

[**0041**] However, if the user of client computer **204** has already participated in the registration process, the log-in process of step **402** may include web server **210** requesting that the user of client device **204** provide his or her previously established user identity and password. As such, the user of computer **204** causes it to transmit his or her user identity and password to web server **210**. Upon receiving them, web server **210** determines their validity. If they are invalid, web server **210** refuses access to client computer **204** wherein flowchart **400** may be discontinued (not shown). However, if the provided user identity and password are valid, the present embodiment proceeds to step **404**.

[**0042**] At step **404**, once the registration process is completed, the present embodiment causes web server **210** to generate a unique user identification (ID) or user key associated with the user of client device **204** that participated in the initial log-in process of step **402**. The unique user ID (or user key) is then stored by web server **210** in a manner that it is associated with that registered user. In this manner, the present embodiment is able to uniquely identify each authorized user of web server **210**. It is appreciated that if a unique user ID has already been generated for a user that logs in with web server **210** at step **402**, the present embodiment may skip step **404** and proceed to step **406**.

[**0043**] In step **406** of **FIG. 4**A, the user of client computer **204** causes it to transmit to web server **210** (e.g., via Internet **202**) a request for a play list of available media content. It is understood that the requested media play list may include all or a portion of the media content available from content server **212**.

[**0044**] At step **408**, in response to web server **210** receiving the play list request, the present embodiment causes web server **210** to transmit to client computer **204** a media content play list together with the unique user ID associated with the logged-in client. The user ID may be attached to the media content play list in a manner such that the user of computer **204** is unaware of it. It is appreciated that the media content of flowchart **400** may include, but is not limited to, high fidelity music, audio, video, graphics, multimedia, and the like. The media content play list of step **408** may be implemented in diverse ways. For example, the present embodiment may generate a media content play list by taking all of the media content titles available from the content server **212** and combining them into a single list. Alternatively, all of the media content titles (or different lists of titles) may be loaded from content server **212** and passed to a Common Gateway Interface (CGI) program operating on web server **210** that may be written in Perl (Practical Extraction and Report Language) Script, where the media titles (or different lists of titles) are concatenated into a single dimensioned array that may be provided to client device **204**.

[**0045**] In step **410** of **FIG. 4**A, the user of client computer **204** may utilize the received media content play list in conjunction with a media player application in order to cause client device **204** to transmit a request to web server **210** for delivery of particular media content, wherein the user ID automatically accompanies the request. The media delivery request of the present embodiment may be implemented in a wide variety of ways. For example, the media content play list provided to client device **204** by web server **210** may enable the user to create one or more customized media play lists by selecting desirable media content titles. It is understood that a customized media play list may establish the media content that eventually will be delivered to client computer **204** along with the sequential order in which it will be delivered. Furthermore, the user of client device **204** may create as many or as few customized play lists as desired and then store them with computer **204**,(e.g., to its desktop) and/or within web server **210**. It is noted that a customized media play list does not actually contain media content, but instead it includes one or more identifiers of specific pieces or portions of media content such as, but not limited to, a song, an audio clip, a video clip, a picture, a graphic picture, a multimedia clip, and the like.

[**0046**] The following Perl Script entitled "User Choice Program" is an exemplary implementation for providing a media play list to enable a user of a computer (e.g., **204**) to generate a request for media delivery by creating one or more customized play lists.

```
#!/usr/bin/perl -w
#-----------------------------------------------------------------------------------------
#                User Choice Program
# This program reads in a mp3 play list source file, displays the file names,
# on a web page to allow for selection, and creates a new play list file to download.
# Programmer: Ted Fitzgerald
# Created for: "themomi.org"
#-----------------------------------------------------------------------------------------
$VERSION = "5.0";
$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';
# ---- decalre some variables used in the code program ------------------ #
# ---- Don't touch these --- #
my $dir_cnt '2 0; #counter
my $qry1 = "";
my $qry2 = "";
my $sq1 = "";
# ------------- YOU CAN/SHOULD LOCALIZE THESE VARIABLES ------------------- #
my $rootdir = "mp3";
my $image_path = "http://www.themomi.org/museum/images";
my $image_url = "www.themomi.org/$rootdir";
my $song_root = "../htdocs/";
my $earthc = "themomi.earthc.net";
# ---------------- Start Code ------------------------------------------ #
$|=1; #flush
print "Content-type: text/html\n\n"; # send basic header
use CGI qw(:standard);
use CGI::Carp('fatalsToBrowser');
use DBI();
use strict;
my $db_host = "AAA.BBB.CCC.DDD";
my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
        "NOBODY", "ASK_FOR_JOE",
        {'RaiseError' => 1}) or die "can't connect to db";
my @play_list;
my $artist;
my $song_name;
print<<HTML_DONE;
    <html>
    <head>
    <META NAME="Organization" CONTENT="www.themomi.org">
    <META NAME="Author" CONTENT="Ted Fitzgerald">
    <META NAME="Quote" CONTENT="Ny hovercraft is full of eels!">
    <META NAME="Description" CONTENT="Music selection / creation software">
    <SCRIPT LANGUAGE="JavaScript">
    <!--
        function changeImages() {
            for (var i=0; i<changeImages.arguments,length; i+=2) {
                document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
            }
        }
// -->
</SCRIPT>
<style><!--a{text-decoration:none}//--></style>
<style><!--a:hover{color:#3399cc; }--></style>
</head>
<body bgcolor="black">
<font size=2 color=white face = arial,san serif, helvetica >
<form method="post">
HTML_DONE
#---------------- Main page display starts here (selection portion ------------- #
# if first time page is displayed submit button hasn't been clicked and submit.blah is undefined.
# we use this parameter to show initial selection or results portion of page
if(! defined(param("Submit.x"))){ #if first time page is displayed submit is undef
print<<HTML_DONE;
    <center>
    <a id="button"> </a>
    <a name="button"> </a>
    <input type="image" value="Submit" src="$image_path/oy.gif" border="0" alt="Create
Playlist" name="Submit">
    <!-- <A HREF="http://www.themomi.org/perl/choose_hi.cgi"
        ONMOUSEOVER="changeImages('oy', 'http://www.themomi.org/museum/images/oy.gif');
return true;"
        ONMOUSEOUT="changeImages('oy',
'http://www.themomi.org/museum/images/oy_over.gif'); return true;">
    <IMG NAME="oy" SRC="http://www.themomi.org/museum/images/oy.gif'
BORDER=0></A> -->
```

-continued

```
    <br><br>
    <input type=Reset value="Reset" name=Reset>
    </center>
HTML_DONE
# ------------------------------ MAIN CODE START HERE ------------------------------ #
my $sql = "select distinct playlist_name from m3u";
    $sql .= " order by playlist_name";
my $sth = $dbh->prepare($sql);
    $sth->execute();
while (my $ref = $sth->fetchrow_hasref()) {
    my $playlist_name = $ref->{'playlist_name'};
    push(@play_list,$playlist_name);
    my $even = $dir_cnt % 2; # flip flop the output to make 2 columns odd = left even = right
    #make left right columns
    if(($even)) {
        print "<td align=center valign=top width=50%>";
    } else {
    print "<table align=center border=0 width=90% bordercolor=black name=outer>"; #start
outer table
    print "<tr><td align= center valign=top widht=50%>";
    }# end if
        print "<br><br><img src=http://$earthc/$image_url/$playlist_name/cover.jpg
align=center><br><br>";
    print "<table border=0 align=center with=90% bordercolor=clock name=internal_table>";
    print "<tr><td><input type=checkbox name=d$dir_cnt></td>";
    print "<td><font size=2 color=white face = arial,san serif, helvetica >";
    print " --- Select All in Category ---</font>";
    print "</td></tr>;
    my $sql1 = "select pl_index,artist,song_name from m3u where playlist_name like
\"$playlist_name\"";
    # print "$sql1<br>";
    my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    while (my $ref1 = $sth1->fetchrow_hashref()) {
        my $id = $ref1->{'pl_index'}; $artist = $ref1->{'artist'}; $song_name = $ref1-
>{'song_name'};
        $artist =~ s/_//g;
        $song_name =~ s/_//g;
        print "<tr><td><input type=checkbox name=\"$id\"></td>\n";
        print "<td><font size=2 color=white face = arial,san serif,helvetica >$artist -
$song_name<br></font></td></tr>\n;
    } #end while ref1
    print "</table>"; # end inner table
    if($even) {
        print "</td></tr>";
        print "</table>"; #end outer table
        print "<center><br><font size=2 color=white face = arial,san serif,helvetica >";
        print "<a href=\"#button\">Return to top</a></font></center>";
    }else{
        print "</td>";
    } #end ending left right column if
    $dir_cnt++; # increment counter used for left/right flip flop
} #end while ref -> display of play lists loop
#-----------------------------------------------------------------
print "<br><br>";
print "<input type=hidden name=size value=$dir_cnt>";
# $dbh->disconnect();
print " </form>";
} #end if (Submit)
#-- done displaying the selection list --
#-----------------------------------------------------------------------------#
# ----------------- DISPLAY RESULTS OF SELECTIONS ------------------------------ #
#
#-----------------------------------------------------------------------------#
if(defined(param("Submit.x")) || defined(param("Submit.x"))){
    my @play_list;
    my @list_copy;
    my @p_list;
    my $pl;
    my $first_pl;
    my $qry1;
    my $qry2;
    #$num_of_dirs '2 param("size");
#
==================================================================
    #my $sql = "select distinct playlist_name from m3u order by playlist_name asc";
```

-continued

```
    my $sql = "select distinct playlist_name from m3u";
        $sql .= " order by playist_name";
    my $sth = $dbh->prepare($sql);
    $sth->execute();
    while (my $ref = $sth->fetchrow_hashref()) {
        my $playlist_nbame = $ref->{'playlist_name'};
        push(@play_list,$playlist_name);
    } #end while "get all playlists" query
#
==================================================================
# ---------------------------------------------------------------------------
# parse param list for "all in playlist" request = "d+some num" & individual songs "num"
# ---------------------------------------------------------------------------
my @song_list;
#my $pl_limit = 5;
#my song.3limit = 50;
#my $pl_count = 1;
foreach (param()) {
    push(@song_list,$_) if / ^ \d+$/;   #just a song number
    # print "<br>$_" if / ^ \d+$/;
    if (/ ^ \d+$/){                #all in playlist d+number
        s/ ^ //l                #remove letter "d"
        push(@p_list,$play_list[$_]);   #store playlist number
        push(@list_copy,$play_list[$_]);
    } # end if "d+numb all playlist "
} # end foreach "item in param() list"
# ---------------------------------------------------------
#           Build "All in playlist" query string
# ---------------------------------------------------------
if(@p_list){
    my $first_pl = shift @p_list;
    my @list_copy = $first_pl;
    $qry1 = "SELECT * FROM m3u WHERE playlist_name IN ( \"$first_pl\"";
    foreach my $pl (@p_list){
#   last if($pl_count = $pl_limit);
        #print "<br>=". @p_list; #print "<br>@p_list";
        $qry1 .= ", \"$pl\" ";
        push(@list_copy,$pl);
#   $pl_count++;
    } #end foreach loop to build items in query string
    $qry1 .= ") order by playlist_name"; #      query ends here
    #print "<br>$qry2";
} #end if anything in p_list array
# ---------------------------------------------------------
#      Build individually selected songs query string
# ---------------------------------------------------------
if(@song_list){ #anyone home in song lists? go for it!
    my @song_copy = @song_list;
    my $first = shift @song_list;
    $qry2 = "SELECT * FROM m3u WHERE pl_index IN ( $first" ;
    foreach my $blarg (@song_list){
        #print "<br>=". @song_list;
        #print "<br>@song_list";
        $qry2 .= ", $blarg ";
    }#
    $qry2 .= ")";
# $qry2 .= " limit $song_limit"; #add limit to number of songs
    #print "<br>$qry2";
} #end if anything in song_list array
# ---------------------------------------------------------
#      Now xreate m3u playlist file
#      Execute query statement(s) and write file
# ---------------------------------------------------------
#print "<br>here's your check of array size greater than zero
defined($qry1),defined($qry2)<br>;
if($qry1 || $qry2){
# ---- Create unique play list name using numbers from date parts --------- #
my $theDate = localtime;
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
my $play_list_name = "Momi_playlist_$2$3.m3u";
# ------------------------------------------------------------------------ #
print "<center><font size=2 color=white face = arial,san serif, helvetica >";
print "<b>Songs in your play list:</b></font></center>;
print " <table align=center>";
open(OUTFILE,">../htdocs/temp/$play_list_name");
print OUTFILE '#EXTM3U' ."\n"; #write file header
```

-continued

```
#print '<br>#EXTM3U<br>'; #write file header
if($qry1){
    my $list = "";
    # now do the playlists
    my $sth1 = $dbh->prepare($qry1);
    $sth1->execute();
    while (my $ref1 = $sth1->fetchrow.3hashref()) {
        my $playlist.3name = $ref1->{'playlist_name'};
        my $pl_header = $ref1->{'pl_header'};
        my $song_url = $ref1->{'song_url'};
        $song_url =~ s/8080/8081/g;
        print OUTFILE "$pl_header\n$song_url\n";
    } # end while
    #print "<b>my list_copy array = @list_copy</b><br>";
    foreach my $list (@list_copy){
        $list =~ s/_//g;
        print "<tr><td><font size=2 color=white face = arial,san serif, helvetica > All Songs from
<b>\"$list\"</b></font></td
></tr>\n";
    } # end foreach
    }#end of $qry1 is defined
if($qry2){
    # now do the selected songs
    my $sth2 = $dbh->prepare($qry2);
    $sth2->execute();
    while (my $ref2 = $sth2->fetchrow_hashref()) {
        my $artist = $ref2->{'song_nam2'};
        my $pl.3header = $ref2->{'pl_header'};
        my $song_url = $ref2->{'song_url'};
        #$print "$pl_header<br>$song_url<br>";
        $artist =~ x/_//g;
        $song_name =~ s/.3//g;
        $song_url '2~ s/8080/8081/g;
        print "<tr><td><font size=2 color=white face = arial,san serif, helvetica >$artist -
$song_name</font></td></tr>";
        print OUTFILE "$pl_header\n$song_url\n";
        } # end while
    }# end if $qry2 is defined
    close OUTFILE;
    print "</table>";
    print "<center><br><br><br>";
    print "<a href=\"http://www.themomi.org/temp/$play_list_name\">";
    print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
    print "</font>";
    print "<br><br><br>";
# ---------------- include text for nubies ---------------------------------- #
print "<br><br>";
print "<table align=center border=0 width=90%>";
print<<MORE;
<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the button above to launch
your playlist<br> with your favorite .mp3/.m3u player
</td>
<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the button above and
use "save target as"<br> to save your playlist file to your computer.</font>
</td></tr>
<tr><td colspan=3 align=center>
MORE
#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
#Bote: the music itself will not be saved locally <br> You will only be downloading one small
.m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>
print "</table><br><br>;
}else{
    print "<br><b>Nothing Selected</b><br>";
    #foreach (param()) {
    #   print "$_<br>";
    #}#end foreach
```

-continued

```
    #print "The value of Param(\"Submit.x"\") is ", param("Submit.x"), "<br>";
    #print "The value of defined Param(\"Submit.x\") is ", defined(param("Submit.x")), "<br>";
    $dbh->disconnect();
    }#end if defined queries
}# end if (Submit.x) "display results of selection "
print "</body>";
print end_html();
#bye bye! Thanks for visiting!
exit(0);
1;
```

[0047] Alternatively, at step **410**, the received media content play list may include a random media content delivery choice that the user of client device **204** may select to transmit a request to web server **210** for delivery of random media content, wherein the user ID automatically accompanies the request. An embodiment for implementing the delivery of media content in a random manner to client computer **204** is described in detail within reference to step **424**.

[0048] At step **412** of **FIG. 4**A, the present embodiment causes web server **210** to determine whether the requesting media application operating on client computer **204** is a valid media application. One of the main functions of a valid media application may be that it is a player of media content as opposed to an application that downloads media content in an unauthorized or unregulated manner. If the web server **210** determines at step **412** that the requesting media application operating on the request client computer **204** is a valid media application, the present embodiment proceeds to step **416**. However, if the web server **210** determines at step **412** that the requesting media application is not a valid application, the present embodiment proceeds to step **414**.

[0049] In step **414**, the present embodiment causes web server **210** to redirect client computer **204** so that it is prevented from accessing the source of the media content. It is understood that the redirection of step **414** may be performed in diverse ways. For example, the present embodiment may cause web server **210** to redirect client computer **204** to a software program that identifies it, logs it out of web server **210** and prevents any subsequent logging in of it for a defined amount of time (e.g., 10 minutes, an hour, a day, a month, a year, or any defined amount of time).

[0050] At step **416**, the present embodiment causes the web server **210** to determine whether the user ID (or user key) that accompanied the media delivery request sent by client computer **204** is valid. If the web server **210** determines that the user ID is not valid at step **416**, the present embodiment proceeds to step **414**. However, if the web server **210** determines that the user ID is valid at step **416**, the present embodiment proceeds to step **418**. It is noted that the order in which steps **412** and **416** are performed by the present embodiment may be switched from the order illustrated within **FIG. 4A**. That is, the present embodiment may perform step **416** before performing step **412**.

[0051] In step **418** of **FIG. 4**A, the present embodiment causes web server **210** to transmit to client computer **204** a redirection command along with a time sensitive access key (e.g., for that day, hour, or for any defined timeframe) thereby enabling the client **204** to eventually receive the requested media content. The redirection command may include a time sensitive address of where the media content is location within content server **212**. The address is time sensitive because periodically the present embodiment of content server **212** renames some or all of the media address directories thereby making previous content source addresses obsolete. It is noted that the actual location of the media content may not change within content server **212**, just the address of the content is changed. However, the locations of the media content may actually be changed along with the corresponding addresses. In any case, unauthorized users and/or applications are restricted from directly retrieving and/or copying the media content from content server **212**. As such, if someone with inappropriate intentions is able to find out where the media content source is stored, the next time he or she tries to use that route (or address) it does not exist anymore thereby preventing further unauthorized access.

[0052] It is noted that within an embodiment of the present invention, the addresses (or routes) of content server **212** that are actively coupled to one or more client devices (e.g., **204**-**208**) are maintained while future addresses (or routes) are being created for new client devices. It is appreciated that as client devices are uncoupled from the media content source of content server **212**, that directory address (or link) may be immediately changed thereby restricting access to unauthorized client devices or applications.

[0053] Alternatively, the functionality of redirection to a media content source may be implemented by utilizing a server network where multiple servers are content providers (e.g., **212**) or by routing a requesting client device (e.g., **204**, **206** or **208**) through multiple servers. Within another embodiment in accordance with the present invention, the delivery of media content from a central content provider (e.g., **212**) may be routed through one or more intermediate servers before being received by the requesting client device (e.g., **204**, **206** or **208**).

[0054] The redirection functionality of step **418** is well suited to be implemented in a wide variety of ways. The following Perl Script entitled "Redirector Program" includes an exemplary implementation for the redirection functionality of step **418** along with other functionality of flowchart **400**. Additionally, this Perl Script records the Internet Protocol (IP) addresses of the client computers (e.g., **204**), the media content requested and its transfer size. In this manner, an implementation of flowchart **400** is able to accurately meter royalty payments, clock usage and transfers, and also keep a log of media content popularity.

```
#!/usr/bin/perl -w
#-----------------------------------------------------------------------------------
#                Redirect Program
#-----------------------------------------------------------------------------------
# This program obscures the proper address of the mp3 source file for due diligence
# purposes and records the requested mp3 file name, requestor ip number, file size
# and date of request to allow for accurate royalty payments & bandwidth monitoring,
# The resulting data is stored in a .csv file for ease of subsequent analysis.
# This program also directs known web browsers to the index page rather than the
# the mp3 file.
#
# Programmer: Ted Fitzgerald
# Created for: "themomi.org"
# ------------------------------------------------------------------------------ #
my $version '2 4.0;
# -----------------------------------------------------------------------------------
$SIG{SHLD} = "IGNORE"; #Skill zombies! Oooo scary scary!;
use IO::Socket::INET;
use DBI();
my $debug = 0; #debugging messages on/off
######################## localize these variables ############################
#
# server.3port: what socket to use = 8080, 8181, 8200, 8282 for hi,mid, low,extra1.low
# mp3_root_dir: the name of the root dir that contains the mp files
#   choices are: changing all the time
# real_dest_url: location of the content server
# send_away_url: location that browsers are sent to if the try to access mp3s.
#
#
#program name
my $level∩   = "hi";            # hi, mid, low, x_lo;
my $short_name = $level . "_redir.cgi";
my $program_name = "http://www.themomi.org/redirs/$short_name";
# logs
my $apache_log = "/usr/local/apach/logs/access_$level.log";
my $error_log = "/usr/local/apache/logs/error_$level.log";
#port
my $server_port '2 8081;
#urls
my $real_dest_url = "www.mpb.tv";
my $send_away_url '2 "www.themomi.org";
#db connection
my $db_host = "AAA.BBB.CCC.DD";
my $db_user = "my user name";
my $db_pw = "some password";
my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                "$db.3user", "$db_pw",
                {'RaiseError' => 1}) or die "can't connect to db";
#rate control songs/min limit
my $limit = 10;
#Changeable access parameters read in from database
my $mp3_root_dir;
my $pass_key;
# change content directory names to prevent direct linking
my $sql_db = "Select die from file_location ";
     $sql1.db .= "where band like \"$level\"";
my $sth_db '2 $dbh->prepare($sql_db);
     $sth_db->execute();
my $ref_db = $sth_db->fetchrow_hashref();
     $mp3_root_dir '2 $ref_db-> {'dir'};
          $sth_db->finish();
     if(!defined($mp3_root_dir)) {
     print "oops!\n";
     #--------------------------------
     # $mp3_root_dir = 37 hi_band";
       $mp3_root_dir = "Fatty";
     # $mp3_root_dir = "ookii";
     #-----------------------------
     # $mp3_root_dir = "mid_band";
     # $mp3_root_dir = "chu";
     # $mp3_root_dir = "normal";
     #-----------------------------
     # $mp3_root_dir = "lo_band";
     # $mp3_root_dir = "chisai";
     # $mp3_root_dir = "Slim_Jim";
     #-----------------------------
```

-continued

```
    # $mp3__root__dir = "x__lo__band";
    # $mp3__root__dir = "chibi";
    # $mp3__root__dir = "Ally__McBeal";
    #-------------------------------
} #end if
my $sql__key = "Select cache__key from cache__key ";
my $sth__key = $dbh->prepare($sql__key);
    $sth__key->execute();
my $ref__key = $sth__key->fetchrow__hashref();
    $pass__key = $ref__key->{'cache__key'};
    $sth__key->finish();
##########################################################################################
my $msg '2 "dubug mode off";
while ($__ = $ARGV[0]) {
    shift;
    #print "found $__ \n";
    last if / ^ --$\;
    if (/ ^ -D/i || / ^ X/i || / ^ -V/i) {
        $msg = "debug mode on";
        $debug = 1;
    }
}#end while
print "\nStarted program: $short__name Version: $version at Port: $server__port $msg \n";
print " Using $real__dest__url/$mp3__root__dir as music source dir\n";
$server = IO::Socket::INET->new(LocalPort=> $server__port,
            Type      => SOCK__STREAM,
            Reuse     => 1,
            Listen    => 100 )
        or die "Couldn't be a top server on port $server__port: $!\n";
#########################################################################
# initialize variables used for player discrimination
my $ok = 1;       # on by default;
my $nsplayer = 0; # this is the windows media player
my $winamp = 0;   # this is the winamp player
my $real = 0;   # this is the real player
my $ripper = 0;   # this is the stream ripper etc.
my $speeder = 0; # this is for speeders
my %what = "";
my $whole__request = "";
my $what__method = "";
my $what__song = "";
my $song.3name = "song";
my $what.3list '2 "list";
my $what__version = "";
my $first__line '2 "";
my $u__a__line = "";
my $ua = "unknown"; #genetic entry
my $counter = 1; #use counter to prevent runaway children
my $bad = 0;
# ------------------------------------------------------------------------------- #
#     Now do main loop. fork off child when connection made
# ------------------------------------------------------------------------------- #
while ($client = $server->accept()) {
    $pid = fork(); #get return value from fork()
    die "Cannot fork: $!" unless defined($pid);
    if ($pid == 0) {
        # Child process starts here
        my $client__info = getpeername($client);
        my $port, my $iaddr) '2 unpack__sockaddr__in($client__info);
        my $remote__host__ip = inet__ntoa($iaddr);
        my $remote__hostname = gethostbyaddr($iaddr, AF__INET); #put in real name
        my $date = localtime;
        if(! defined($remote__hostname) ){ $remote__hostname = "unknown";}
        if($remote__host__ip =~ /10.10.0.252/o || $remote__host__ip =~ /10.10.0.251/o){
            exit(1);
        }
        if($debug){
        print "\n ----------------------------------------------------------------------------\n";
        print "$short__name: client ip '2 " . $remote__host__ip . " client__hostname = " . $remote__hostname . "\n";
        }
#evaluate the first line
        $what = <$client>;
#ignore network health check from x.x.x..104
        if(! defined($what) && $remote__host__ip !~ /209.10.35.104/){
            print "$short__name@$server__port encountered a client who said noting! $remote.3host__ip , $date<-- \n "
;
```

-continued

```
            exit(1);
        }
        $first_line = $what;
        if($debug){ print "\n First line is \"", $what , "\" \n"; }
        if ($what !~ / ^ (GET|HEAD)\s+/i) { #valid requests must start with GET or HEAD
            if($remote_host_ip !~ /209.10.35.104/o){
            print "$short_name -> bad req HTTP/1.1 400, no GET HEAD, $remote_host_ip said:->$what<- \n";
            }
            print $client 'HTTP/1.1 400, "Bad Request"';
            exit(1);
        }
        if ($what =~ /$mp3_root_dir/i) {
            print "$short_name -> bad req HTTP/1.1 400, mp3_dir in request, $remote_host_ip said:->$what<- \n";
            print $client 'HTTP/1.1 400, "Bad Request"';
            exit(1);
        }
#banned ip list
        if ($remote_host_ip =~/132.239.12.77/
            || $remote_host_ip =~ /66.122.240.11/
            || $remote_host_ip =~ /63.17.233.236/) {
            #print " $short_name sending HTTP/1.1 400, baned $remote_host_ip \n said:->$what<- \n";
            print $client 'HTTP/1.1 400, "Bad Request"';
            exit(1);
        }
        #now parse line, extract what we need or dump 'em
        if ($what =~ / ^ ([A-Z]+)\s+([\S]+)\s+HTTP\/([\d\.]+)\s*/){
            $whole_request = $what;
            $what_method = $1;
            $what_song = $2;
        $what_verwsion = $3;
        }else{
            print " $short_name: sending bad client HTTP/1.1 400, Bad Request message\n";
            print " $short_name: bad req was: $what\n;
            print $client 'HTTP/1.1 400, "Bad Request"';
            exit(1);
        }
        # print "method is > $what_method< song is >$what_song< version is >$what_version< \n";
        if ($what_method !~ /(GET|HEAD)/i) {
            exit(1);
        }
        #now loop to get other info lines
        while ($request '2 <$client>) ne ("\r\n\r\n") ){
        last if(! defined($request));
        $request =~ s/[\r\n]//g;
        if($debug) {
            print "<level: $counter >The client said: " . $request . "<\n";
        }
        #----- WEB BROWSER INFO ------ #
        if($request =~ /Mozilla/io) {
                $ok = 0; $mozzy = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        }
        if($request =~ /Opera/io) {
                $ok = 0; $mozzy = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        ◇
        if($request =~ /Omniweb/io) {
                $ok = 0; $mozzy = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        }
        if($request =~ /iCab/io)   {
                &ok '2 0; $mozzy = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        }
        #----- MP3 RIPPER INFOR ------ #
        if($request =~ /StreamRipper/io) {
        $ok = 0; $ripper = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        }
        #----- No get for Wget ------ #
        if($request =~ /Wget/io) {
                $ok '2 0; $ripper = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        }
        #----- FlashGet fizzles ------ #
        if($request =~ /FlashGet/io) {
                $ok = 0; $ripper = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        }
        #----- GetRight is Wrong ------ #
        if($request =~ /GetRight/io) {
                $ok '2 0; $ripper = 1; $request =~/.*:\s*(.+)$/; $ua = $1;
        }
```

-continued

```
        #----- Step on Spiders ------ #
        if($request =~/asterias/io) {
                $ok = 0; $ripper = 1; $reuquest =~/.*:\s*(.+)$/; $ua = $1;
        }
        #----- Web access = No access ------ #
        if($request =~ /Web access/io) {
                $ok = 0; $ripper = 1; $request =~/.*:\s*(.+)$/; $ua = $1;
        }
        #----- Monica gets the bone ------ #
        if($request =~ /monica/io) {
                $ok = 0; $ripper = 1; $request =~/.*:\s*(.+)$/; $ua = $1;
        }
        #----- man what the hell is this? ;-) ------ #
        if($request =~ /Media Jukebox WinInet Reader/io){
                $ok = 0; $ripper = 1; $request =~/.*:\s(.+)$/; $ua = $1;
        }
        #----- burn Nero Player ------ #
        if($request =~ /NeroMediaPlayer/io) {
                $ok = 0; $ripper = 1; $request =~/.*:\s*(.+)$/; ua = $1;
        }
        #----- ISMUSIC is deaf ------ #
        if($request =~ /1SMUSIC/io) {
                $ok = 0; $ripper = 1; $request =~/.*:\s*(.+)$/; $ua = $1;
        }
        #----- I know who Anonymizer is ------ #
        if($request =~ /Anonymizer/io) {
                $ok = 0; $ripper = 1; $request =~/.*:\s*(.+)$/; $ua = $1;
        }
        #----- MP3 CLIENT INFO ------ #
        if($request =~ /NSPlayer/io) {
                $ok = 0; $nsplayer = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        }
        if($request =~ /RMA/io)   {
                $ok '2 0; $real = 1; $request =~/.+:\s*(.+)$/; $ua = $1;
        }
        if($request =~ /Winamp/io)   { $ok = 1; $request =~/.+:\s*(.+)/s*$/; $ua = $1; }
        if($request =~ /QuickTime/io)   { $ok = 1; $request =~/.+:\s*(.+)/s*$/; $ua = $1;}
        if($request =~ /Xaudio/io)   { $ok = 1; $request =~/.+:\s*(.+)/s*$/; $ua = $1;}
        if($request =~ /AppleApp/io)   { $ok = 1; $request =~/.+:\s*(.+)/s*$/; $ua = $1;}
        if($request =~ /iTunes/io)   { $ok = 1; $request =~/.+:\s*(.+)/s*$/; $ua = $1;}
        if($request =~ /Sonique/io)   { $ok = 1; $request =~/.+:\s*(.+)/s*$/; $ua = $1;}
        if($request =~ /xmms/io)   { $ok = 1; $request =~/.+:\s*(.+)/s*$/; $ua = $1;}
        if($request =~ /Uplayer/io)   { $ok = 1; $request =~/.+:\s*(.+)/s*$/; $ua = $1;}
        #=== hold for error repotin ===#
        if($request =~ /User Agent/io)   { $u_a_line = $request; }
        last if ($counter > 20 ); #we don't wnat some client babbling on and on
        $counter++;
    }#end while
if($debug){
    print "User Agent = $ua \n";
    print "request = $what_song \n";
    }
                    # ###### Rate controls to prevent excessive access to content
    $remote_host_ip =~ /.+.(.+)/;
    # my $last_octet_mod = $1;
    # $last _octet_mod = $last_octet_mod % 1; #mod by number of tables we want
    # my $time_offset '2 1;
    my $sql0 = "Select count(ip_address) as loser_count from loser ";
    $sql0 .= "where ip_address = \"$remote_host_ip\" ";
    $sql0 .= "AND user_agent = \"$ua\"";
    my $sth0 = $dbh->prepare($sql0);
        $sth0->execute();
    my $ref0 = $sth->fetchrow_hashref();
    my $loser_count = $ref0->{'loser_count'};
        $sth0->finish();
# #
    if($loser_count <= 0 ){
    my $sql '2 "SELECT COUNT(request_time) as number_songs_reqs ";
        #$sql .= "MINUTE(MIN(request_time)) - MINUTE(now()) as duration ";
        $sql .= "FROM rate_control_0 ";
        $sql .= "WHERE ";
        $sql .= "ip_address = \"$remote_host_ip\" ";
        $sql .= "AND ";
        $sql .= "user_agent = \"$ua\" ";
        $sql .= "AND ";
        $sql .= "request_time > now() - INTERVAL 60 SECOND";
```

-continued

```
    print "\n$sql\n" if $debug;
    my $sth = $dbh->prepare($sql);
        $sth->execute();
    my $ref = $sth->fetchrow_hashref();
    my $song_count = $ref->{'number_songs.3reqs'};
    #my duration = ($ref->{'duration'});
        $sth->finish();
        print "\n Song count is $song_count \n " if $debug;
        if($song_count >= $limit){
        $speeder =1;
        $ok = 0;
        print "\n Loser exceeded limit\n" if $debug ;
        my $sql1 = "insert into loser (ip_address,user_agent,loser_time) values (\"$remote_host_ip\",\"$ua\",now()
)";
        my $sth1 = $dbh->prepare($sql1);
        $sth1->execute();
        $sth1->finish();
    }#end if song_count
    }else{
        print "\n loser is in loser table\n" if $debug;
        $speeder = 1;
        $ok = 0;
    }
    if(!$speeder){
    my $sql1 = "INSERT INTO rate_control_0 ";
        $sql1 .= "(ip_address,user_agent,request_time) ";
        $sql1 .= "VALUES (\"$remote_host_ip\",\"$ua\",now())";
        my $sth1 = $dbh->prepare($sql1);
    $sth1->execute();
    $sth1->finish();
    }#end if
    print $client "HTTP/$what_version 302 Found \r\n";
    print " I told him HTTP/$what_version 302 Found \n" if $debug;
    print $client "Date: $date \r\n";
    print $client "Server: Apache/1.3.20 \r\n"; #lie to 'em to keep 'em guessing.
    print $client "Keep-Alive: timeout=10, max=20\r\n";
    #------------- Good guys -------------- #
    if(($nsplayer || $real) && ! $speeder){
    print $client "Location: http://$real_dest_url:80/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    #   print $client "Location: http://$real_dest.3url:80/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url:80/$mp3_root_dir$what_song?arf=$pass_key\r\n" if $debug;
    print "found nsplayer or real player\n" if $debug;
    }
    if($ok && ! $speeder){
    print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song?arf=1$pass_key\r\n";
    # print $client "Location: http://$real_dest_url/$mp3_root_dir$what_song\r\n";
    print "\nLocation: http://$real_dest_url/$mp3_root_dir$what_song?arf=$pass_key\n" if $debug;
    # print "other player that doesn't need to be told what port to go to\n";
    }
    #------------- Bad guys --------------- #
    if($mozzy){
    print $client "Location: http://$send_away_url/index.html\r\n";
    #   print "mozzy type web server being redirected to index page\n";
    }
    if($ripper){
    print $client "Location: http://$send_away_url/ring_of_hell/Pentiums.mp3\r\n";
    # print "mozzy type web server being redirected to index page\n";
    }
    if($speeder ){
        print "$short_name: Speeder @ $remote_host_ip sending bad client HTTP/1.1 400, Bad Request
message\n";
        print "$short_name: Speeder said $what\n";
        print $client 'HTTP/1.1 400, "Bad Request"';
    }
    $ok = 1;
    $nsplayer = 0;
    $mozzy = 0;
    $real '2 0;
    print $client "\r\n\r\n";
    # print "\ndone\n";
    # print "*********************************************************\n\n";
#   $dbh->disconnect();
    close $client;
    $what_song =~ /\/(.+)\/(.+).mp3/io;
    $what.3list = $1;
    $song_name = $2;
```

-continued

```
    #trash filter ************************************************************
    if(! defined($what_list)){
            print "$short_name: no list\n"; $what_list = "none\n";
    }
    if(! defined($song_name)){
            print "$short_name: no song_name $remote_host_ip\n"; $song_name= "none";
            $bad = 1;
    }
    if(! defined($remote_hostname)){
            print "$short_name: no hostname\n"l; $remote_hostname = "none";
            $bad = 1:
    }
    if(! defined($remote_host_ip)){
            print "$short_nameL: no host_ip\n"; $remote_host_ip = "none";
            $bad = 1;
    }
    if(! defined($remote_hostname)){
            print "$short_name: no hostname $remote_host_ip\n"; $remote_hostname = "none";
            $bad = 1;
    }
    if(! defined($ua)){
            print "$short_name: no ua\n"; $ua = "none";
            $bad = 1;
    }
    #match this: APACHE LOG FORMAT (%h %l %u %t \"%r\" %s %b \"%{Referer}i\" \"%{User0agent}i\")
    #change date from: Fri Nov 16 16:33:26 2001 => [16/Nov/2001:16:32:33 -0500]
        $date =~ /.+\s(.+)\s(.+)\s(.+)\s(.+)/io;
        my $month_name = "$1";
        my $day = "$2";
        my $time = "$3";
        my $year = "$4";
        if($day < 10){ $day = "0$day"; } #hack to add leading zero
    $month_name =~ s/ //g;
        #my $apache_date = "[day/$1/$4:$3 -0500]";
        my $apache_date = "[$day/$month_name/$year:$time -0500]";
        $whole_request =~ s/\r//;
        $whole_request =~ s/\n//;
        open (APACHE_LOG, ">>$apache_og") or warn "Waaaa! Can't open $apache_log";
        $a_str = "remote.3host_ip - - $apache_date \"$whole_request\" 302 2 \"$program_name\" \"$ua\"";
        print APACHE_LOG "$a_str\n";
        close APACHE_LOG;
        if($bad || $ua eq "unknown" ){
        $first_line =~ s/[\r\n]//g;
        open (Error_LOG,">>$error_log");
        my $e_str = "$remote_host_ip - - $apache_date \"$first_line\" 400 2 \"$short_name\" \"$u_a_line\"";
        print ERROR_LOG "$e_str\n";
        close ERROR_LOG;
        }#end if error
        close STDOUT;
        close STDERR;
        close STDIN;
        exit(0);   # Child process return status and exits when done.
    } # else 'tis the parent process, which goes back to accept()
}
$dbh->disconnect();
close ($server);
exit(0);
```

[0055] At step **420** of **FIG. 4**B, upon receiving the redirection command, the present embodiment causes the media application operating on client computer **204** to automatically transmit to content server **212** a new media delivery request (that automatically includes the time sensitive access key) that includes the address of the desired media content.

[0056] In step **422**, the present embodiment causes content server **212** to determine whether the time sensitive access key associated with the new media delivery request is valid. If the content server **212** determines that the time sensitive access key is not valid at step **422**, the present embodiment proceeds to step **414** of **FIG. 4A**. However, if the content server **212** determines that the time sensitive access key is valid at step **422**, the present embodiment proceeds to step **424**.

[0057] At step **424** of **FIG. 4**B, the present embodiment causes content server **212** to transmit the requested high fidelity media content to client computer **204**. It is noted that when content server **212** transmits the media content to client computer **204** it does not encrypt the data stream. Additionally, content server **212** transmits the media content in a burst load (rather than a fixed data rate) that transfers to client device **204** as fast as its network transfer rate allows. The present embodiment may cause content server **212** to adapt its download speed to whatever network transfer rate

the client computer's is able to handle. For example, if client computer **204** is coupled to the Internet **202** via a T1 communications line, content server **212** transfers the media content to client computer **204** at transmission speeds allowed by the T1 communications line. As such, once the requested high fidelity media content is transmitted to client computer **204**, content server **212** is then able to transmit requested high fidelity media content to another client computer (e.g., **206** or **208**). It should be appreciated that by performing step **424** in this manner, the delivery of high fidelity media content by the present embodiment becomes very efficient in terms of a statistical distribution over time and does not overly congest the utilized communication network(s).

[**0058**] It is appreciated that the delivery of the media content to client device **204** by content server **212** at step **424** may be implemented in a wide variety of ways. For example, while delivering the media content to client computer **204** at step **424**, a rate control restrictor functionality may be resident to content server **212** in order to monitor and limit "suspicious" media content retrieval. For instance, if a client device (e.g., **204**) tries to retrieve media content from content server **212** faster than a predefined limit, the rate control restrictor uncouples client device **204** from its media content source (e.g., content server **212**) and continues to restrict its access from the source for a predefined amount of time, as described herein. One of the reasons for implementing this rate control restrictor is to restrict access to those unauthorized applications that are designed to retrieve and/or copy media content from the source in an unauthorized manner. For example, a client device (e.g., **204**) may be retrieving pieces of audio content at a rate determined to be much faster than is humanly possible to listen to in real time. As such, the rate control restrictor (which may be implemented with software and/or hardware) uncouples the client device from the media content source (e.g., content server **212**) and restricts its access for a predefined amount of time.

[**0059**] Alternatively, an embodiment of the present invention may be implemented such that a user of a client device (e.g., **204**) may establish a huge buffer of media content (e.g., audio clips, video clips, and the like) to be delivered by content server **212**. However, the present embodiment at step **424** may just provide client computer **204** one piece of media content at a time as would be typically needed to utilize (e.g., listen to, watch, etc.) that piece of media content in real time.

[**0060**] Within another embodiment in accordance with the present invention, embedded keys and/or digital watermarks may be implemented within media content stored by content server **212** which may be delivered during flowchart **400** of **FIGS. 4A and 4B**. By using embedded keys and/or digital watermarks within the media content, it is easier to determine if some unauthorized person has been retrieving and/or copying media content from a media content source of content server **212**. The embedded keys and/or digital watermarks within media content may include, but are not limited to, information indicating where the media came from, the identity of the media requestor and/or the identity of the requestor client device (e.g., **204**, **206** or **208**). With this information, it may be determined by the present embodiment who or what client device has been unauthorized in retrieving and/or copying media content from one or more of the media sources (e.g., **212**). As such, that person and/or client device (e.g., **204**, **206** or **208**) can be permanently restricted by the present embodiment from requesting and/or accessing media content.

[**0061**] It is understood that if the user of client computer **204** caused it at step **410** to transmit to web server **210** a request for delivery of random media content, this request information may be communicated to content server **212** in order for it to fulfill the request at step **424**. The delivery of media content in a random manner at step **424** may be implemented in diverse ways in accordance with the present embodiment. For example, a random media play list may be generated by first taking all of the titles of the media content stored by content server **212** and concatenated them into a single dimensioned array. Next, a random number may be generated. It is appreciated that a random number may be generated in a wide variety of ways in accordance with the present embodiment. For instance, a random number generator may incorporate as its product the combination of:

[**0062**] 1. A number produced by a random function. It is noted that the random number may not truly be a random number, but a random sequence that follows a "seeded" number. This seeded number is a starting point for the further permutations of random selection, but would have been predictable with a fixed seed value.

[**0063**] 2. The Julian date maintained within content server **212** that includes the hours, minutes, seconds, day, date, and four-digit year.

[**0064**] Using this combination, the following code may then be executed:

```
$theDate = 'date'; #get system date
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
$play_list_name = "MoMI_random_playlist_$2$3.m2u";
$random_seed = "$3$2$3"; #yse time parts to make random
srand $random_seed; #seed the random num gen.
```

[**0065**] It is appreciated that the time/date provides a basis for seeding the random number generator that is repeatable once per millennium. Now that a random number has been selected, the actual process of applying this to the array of media content titles follows.

[**0066**] Exemplary criteria for selecting a media content title for inclusion in the random play lists may be, but is not limited to, three-fold:

[**0067**] 1. There cannot be more than 50 media content titles total in the play list;

[**0068**] 2. The selection should come from no more than 20 play lists stored by content server **212**; and

[**0069**] 3. The selection should not have been previously selected.

[**0070**] Assuming that the first two conditions have not been met, the resulting random number is used as the index into the media content title array and selects the song title that resides in that location in the array. If that title has been previously selected, a new random number is requested, and the process continues until all 3 conditions are fulfilled.

[0071] The resulting list of media titles are then formatted into a useable play list (or what may be referred to as a M3U file), and it is written to a memory device of content server **212** while a page is prepared for presenting this information to the user of client computer **204**.

[0072] The following Perl Script entitled "Random Play List Program" is an exemplary implementation for enabling content server **212** to provide a random play list of media content as described herein.

```
#!/usr/bin/perl -w
#-----------------------------------------------------------------------------
#
#                RANDOM PLAY LIST PROGRAM
#
# This program reads in a mp3 play list source file in many directories ,
# displays the random choice of file names ,
# and creates a new file to download.
# Programmer: Ted Fitzgerald
# Created for "themomi.org"
#-----------------------------------------------------------------------------
my $version;
$version '2 5.0;
$ENV{'PATH'} = '/bin:/usr/bin:/usr/local/bin';
# declare variables
my ($i,$j); #just counters
my ($play_list_name); #caribale to create random file name for download
my ($how_many_songs); # = total number of songs in all the dirs.
my (@thename); #array to contain the name of the randomly selected song
my (@thepath); #array to contain the URL path of the randomly selected song
my ($showname); #used to find the portion of name we want to display.
my ($song_url);
my (@theheader);
my (@artist);
my ($artist);
my ($theDate);
# ---------------------------- user adjustable parameters ----------------------------
my $max_num_songs = 34; # set limit of how many songs to display
my $maxdirs = 20; #set limit on number of dirs to look in
my $mp3dir = "../htdocs/mp3"; #path from cgi script to mp3 directory
my $center_col_size = ($max_num_songs /2 + 1);
$|=1; #flush the buffer
use strict;
use CGI qw(:standard);
use CGI::Carp('fatalsToBrowser');
use DBI();
my $db_host = "AAA.BBB.CCC.DDD";
my $dbh = DBI->connect("DBI:mysql:database=mpb;host=$db_host",
                "PIZZA_MAN", "HE_DELIVERS",
                {'RaiseError' => 1}) or die "con't connect to db";
my $q = new CGI;
print "Content-type: text/html\n\n";
print "<html>";
print "<head>";
print "<title>Random Play List Creation</title>";
print "<style><!--a{text-decoration:none}//--></style>";
print "<style><!--a:hover{color:#3399cc; }--></style>";
print <<HTML_DONE;
<SCRIPT LANGUAGE="JavaScript">
<!--
function changeImages() {
    for (var i=0; i<changeImages.arguments.length; i+=2) {
        document[changeImages.arguments[i]].src = changeImages.arguments[i+1];
    }
}
// -->
</SCRIPT>
<head>
<body bgcolor="black" topmargin="0" leftmargin="0" marginheight="0" marginwidth="2"
text="white" link="#ffcc66" vlink="#ffcc
66" alink="#ffcc66">
<font size=2 color=white face = arial,san serif, helvetica >
HTML_DONE
$theDate = localtime; #get system date
$theDate =~ /(\d+):(\d+):(\d+)\b/i;
$play_list_name = "Momi_random_playlist_$3$2$3.m3u";
#-----------------------------------------------------------------------------#
# do all file io before displaying the names in list
my $sql = "select pl_header,song_name,song_url,artist from m3u order by rand() limit
```

-continued

```
$max_num_songs";
    my $sth = $dbh->prepare($sql);
        $sth->execute();
    while (my $ref ' $sth->fetchrow_hashref()) {
        my $pl_header = $ref->{'pl_header'};
        my $song_name = $ref->{'song_name'};
        my $song_url = $ref->{'song_url'};
        my $artist = $ref->{'artist'};
        push (@thename,$song_name);
        push (@thepath,$song_url);
        push (@theheader,$pl_header);
        push (@artist,$artist);
    }
open(OUTFILE,">../htdocs/temp/$play_list_name");
#write out file header
print OUTFILE "#EXTM3U\n";
for($i=0;$i<$max_num_songs;$i++){
    print OUTFILE "$theheader[$i]\n";
    $song_url = "$thepath[$i]";
    $song_url =~ s/8080/8081/io;
    print OUTFILE "$song_url\n";
}#end for
close OUTFILE;
#-----------------------------------------------------------------------------#
#now show 'em what they have in the list
print<<MORE;
<center>
MORE
print "<br>";
print "</center>";
print "<table align=\"center\" border=0 width = \"95%\">";
print "<tr width=45%><td><td rowspan=$center_col_size align=center valign=top>";
print <<HERE;
<A HREF="http://www.themomi.org/temp/$play_list_name"
        ONMOUSEOVER="chaneImages('create_now_vert_01',
http://www.themomi.org/museum/newimages/create_now_vert_over.gif');
return true;"
        ONMOUSEOUT="changeImages('create_now_vert_01',
http://www.themomi.org/museum/newimages/create_now_vert.gif); return
true;">
        <IMG NAME="create_now_vert_01"
SRC="http://www.themomi.org/museum/newimages/create_now_vert.gif" BORDER=0></A>
HERE
print "</td>";
print "<td width=45%></td></tr>";
for($i=0;$i<$max_num_songs;$i++){
    my $flip_flop = ($i % 2 );
    $showname = $thename[$i];
    #$showname =~ /EXTINF:\d+,(.+)/i;
    $showname =~ s/_//g;
    $artist = $artist[$i];
    $artist =~ s/_//g;
    if ($flip_flop){
    print "<td width = 45% align=\"center\"><font size=2 color=white face = arial, san serif,
helvetica >$artists - $showname
</font></td></tr>;
    }else{
    print "<tr><td width= 45% align=\"center\"><font size=2 color=white face = arial,san serif,
helvetica >$artist - $showna
me </font></td>";
    }
}
print "</table><br>";
#-----------------------------------------------------------------------------#
# print "<center>";
# print "<br><a href=\"http://www.themomi.org/temp/$play_list_name\">";
# print "<img src=\"http://www.themomi.org/museum/newimages/create_now.gif\"
border=0></a>";
# print "</center>";
print "<table align=center border=0 width=90%>";
print<<MORE;
<tr><td width=40% align=center> </td>
<td align=center with=10%><font face="Arial,Helvetica,sans-serif" color="gold" size="4">
MORE
print "<A HREF=\"http://www.themomi.org/perl/random_hi.cgi\"";
print<<MORE;
```

-continued

```
    ONMOUSEOVER="changeImages('rerandom__now__vert__01',
http://www.themomi.org/mudeum/newimages/re-randomize__sm__over./gif');
    return true;"
ONMOUSEOUT="changeImages('rerandom__now__vert__01',
'http://www.themomi.org/museum/newimages/re-randomize__sm.gif');
    return true;">
        <IMG NAME="rerandom__now__vert__01"
SRC="http://www.themomi.org/museum/newimages/re-randomize__sm.gif" BORDER=0></A>
</font></td>
<td> </td>
</tr>
<tr><td width=40% align=center>
<font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Left click<br> (Mac: single click)</b><br> on the center button above to launch
your <br>playlist with your favorite .mp3/.m3u player
</td>
<td align=center width=10%><font face="Arial,Helvetica,sans-serif" color="gold" size="4">--
OR --</font></td>
<td align=center width=40%><font face="Arial,Helvetica,sans-serif" color="#6699cc" size="3">
<b>Right click<br> (Mac: ctrl + single click)</b><br> on the center button and
use "save target as"<br> to save your playlist file to your computer.</font>
</td></tr>
<tr><td colspan=3 align=center>
MORE
#<font size="2" face="Arial,Helvetica,sans-serif" color="white"><br>
#Note: the music itself will not be saved locally <br> You will only be dowloading one small
.m3u file which will
#play the songs from MPB.TV whenever you wish</font><br>
#</td></tr>
print "</table><br><br>";
print "<center>";
print $q->end_html;
1;
```

[0073] In step **426** of **FIG. 4**B, upon receiving the requested high fidelity media content from the content server **212**, the present embodiment causes client computer **204** to store it. The present embodiment may cause client device **204** to store the received media content in a manner such that it is not easy for the user of client computer **204** to redistribute the media content in an unauthorized manner. For example, the present embodiment may cause the high fidelity media content to be stored by a memory device of client device **204** utilizing one or more hidden directories where it may be cached for a limited amount of time. As part of restricting access to the media content, the present embodiment at step **412** may reject any media player application if it allows accessibility to one or more hidden directories. It is noted that if the user of client computer **204** turns it off or quits out of the media player application, the stored media content usually is typically deleted from the memory of client **204**.

[0074] Alternatively, at step **426** the present embodiment may cause client device **204** to store the received media content in a non-temporary manner within the media application operating on it or within one of its Internet browser applications (e.g., Microsoft Internet Explorer™, Netscape Communicator™, and the like) such that the media content may be utilized in a repetitive manner. Additionally, the received media content may be stored in a manner such that it is not easy for the user of client computer **204** to redistribute it in an unauthorized manner. For example, the stored media content may be implemented in a way such that it may be utilized by client device **204**, but it may be restricted from being download to another client device (e.g., **206** or **208**). It is noted that by storing the media content within client device **204** in this manner, content server **212** is relieved from redelivering the same media content to client device **204** each time its user desires to utilize (e.g., listen to, watch, etc.) it. Therefore, by not having to redeliver media content to client devices (e.g., **204-208**), it is appreciated that flowchart **400** may be even more efficient in delivering media content to client devices.

[0075] At step **428**, the user of client computer **204** may cause the media application operating on it to access and utilize the delivered high fidelity media content thereby enabling its user or users to experience (e.g., listen to, view, etc.) the media content. It is noted that a specialized or proprietary media player is not needed in order to utilize the received media content. Instead, an industry standard media player may be utilized by client computer **204** to utilize the received media content. Some of the exemplary industry standard media players that are typically available at no cost and are supported by longstanding organizations may include, but are limited to, Windows™ Media Player™ for personal computers (PCs), iTunes™ Player or Quicktime™ for Apple computers, and XMMS Player for computers utilizing the Linux operating system.

[0076] **FIG. 5** is a diagram of an exemplary global media content delivery system **500** in accordance with an embodiment of the present invention. System **500** may be utilized to globally deliver media content (e.g., audio, video, graphics, multimedia, etc.) to client devices (e.g., **204**, **206** and/or **208** of **FIG. 2**) in conjunction with any manner similar to that described herein. System **500** includes a global delivery network **502** that may include multiple content servers (e.g., **504**, **506**, **508**, **510**, **512**, **514** and **516**) located throughout the world which may be referred to as "points of presence."

Each of content servers **504-516** may store a portion, a substantial portion or the entirety of a media content library that may be provided to client devices via a network, such as, the Internet (e.g., **202**) or a wide area network (WAN). As such, each of content servers **504-516** may provide media content to client devices in its respective vicinity of the world.

[**0077**] For example, a content server (e.g., **516**) located in Tokyo, Japan is able to provide media content from its stored media content library to client devices (e.g., **204**, **206** and **208**) within the Asian part of the world while a content server (e.g., **512**) located in New York City is able to provide media content from its stored media content library to client devices within the eastern United States and Canada. It is noted that each city name (e.g., Hamburg, San Jose, Amsterdam, New York City, London or Tokyo) associated with one of the content servers **506-516** represents where that particular content server is located. However, these city name are exemplary since content servers **504-516** may be located anywhere in the world and are not limited to the locations shown within system **500**.

[**0078**] It should be understood that system **500** is described in conjunction with **FIGS. 2, 3** and **4** in order to more fully describe the operation of the present embodiment. Specifically, after a client device (e.g., **204**) interacts with a central web server (e.g., **210**) as described herein, the web server **210** may then redirect client device **204** to receive the desired media content from a content server (e.g., **504**, **506**, **508**, **510**, **512**, **514** or **516**) based on one or more different criteria.

[**0079**] For example, client device **204** may be located in Charlotte, N.C. and its user causes it to log-in with a central web server **210** (which may be located anywhere in the world). It is understood that steps **402-416** of **FIGS. 4A** may then be performed as described herein such that the present embodiment proceeds to step **418** of **FIG. 4B**. At step **418**, the present embodiment may then determine which content server (e.g., **504-516**) may subsequently provide the desired media content to client computer **204**. The present embodiment may use one or more different criteria in determining which content server to choose. For example, the present embodiment may base its determination on which content server is located the shortest distance from client device **204**. This may be performed by utilizing stored registration information (e.g., address) provided by the user of client computer **204**. Alternatively, the present embodiment may base its determination on which content server provides media content to the part of the world that client device **204** is located within. However, if each of the content servers (e.g., **504-516**) do not store the same media content, the present embodiment may base its determination on which content server can actually provide the media content desired by client computer **204**. It is noted that these are exemplary determination criteria and the present embodiment is not limited to such implementation.

[**0080**] Once the present embodiment determines which content server (e.g., **512**) is to provide media content to client computer **204** at step **418**, the present embodiment then causes web server **210** to transmit to client computer **204** a redirection command to content server **512** along with a time sensitive access key (e.g., for that day, hour, or for any defined timeframe) thereby enabling the client **204** to eventually receive the requested media content. Within system **500**, the redirection command may include a time sensitive address of where the media content is location within the content server **512**. As such, the New York City content server **512** may subsequently provide the desired media content to client device **204**. It is noted that steps **418-428** and **414** of **FIG. 4B** may then be performed with content server **512** in a manner similar to content server **212** as described herein.

[**0081**] One of the advantages of utilizing multiple content servers (e.g., **504-516**) to provide high fidelity media content to client computers (e.g., **204-208**) throughout the world is that communication network systems of the Internet do not become overly congested. Additionally, global network **502** is able to deliver media content to a much larger amount of client computers (e.g., **204-208**) in a more responsive manner.

[**0082**] Within **FIG. 5**, it is understood that content servers **504-516** of the global delivery network **502** may be coupled in a wide variety of ways in accordance with the present embodiment. For example, content servers **504-516** may be coupled utilizing wired and/or wireless communication technologies. Additionally, it is noted that content servers **504-516** may be functionally coupled such that if one of them fails for some reason, another content server may take over and fulfill its functionality. Furthermore, one or more web servers similar to web server **210** may be coupled to the global delivery network **502** utilizing wired and/or wireless communication technologies.

[**0083**] Within system **500**, content server **504** includes a web infrastructure that is a fully redundant system architecture. It is understood that each content server (e.g., **506-516**) of the global delivery network **502** may be implemented to include a web infrastructure in a manner similar to that implemented for content server **504**.

[**0084**] Specifically, the web infrastructure of content server **504** includes firewalls **518** and **520** which are each coupled to global network **502**. The firewalls **518** and **520** may be coupled to global network **502** in diverse ways in accordance with the present embodiment. For example, firewalls **518** and **520** may be coupled to global network **502** utilizing wired and/or wireless communication technologies. More specifically, firewalls **518** and **520** may each be coupled to global network **502** via a 10/100 Ethernet handoff. However, system **500** is not in anyway limited to this particular implementation. It is understood that firewalls **518** and **520** are implemented in order to prevent malicious users from accessing in an unauthorized manner any part of the web infrastructure of content server **504**, especially its stored media content. Firewalls **518** and **520** are coupled together. Furthermore, firewall **518** includes a device **536** and a DB server **540** while firewall **520** includes a device **538** and a DB server **542**. Additionally, device **536** is coupled to firewall **518**, DB servers **540** and **542**. Moreover, device **538** is coupled to firewall **520**, DB servers **542** and **540**.

[**0085**] Within **FIG. 5**, firewall **518** is coupled to a director device **522** which is coupled to an internal web application server **526**, a hub server **530** and an internal web application server **528**. The firewall **520** is coupled to a director device **524** which is coupled to internal web application servers **528** and **526**, a staging server **534** and hub server **530**. The hub server **530** may be implemented in diverse ways in accor-

dance with the present embodiment. For example, hub server **530** may be specifically implemented as a Linux hub server. The hub server **530** is coupled to a storage device **532** capable of storing media content. Storage device **532** may be implemented in a wide variety of ways in accordance with the present embodiment. For example, storage device **532** may be implemented as a RAID (redundant array of independent disks) appliance.

[**0086**] It is understood that content servers **504-516** may each be implemented in any manner similar to content server **212** described herein. Furthermore, content servers **504-516** of the present embodiment may each be implemented as one or more physical computing devices (e.g., computer **100** of **FIG. 1**).

[**0087**] Accordingly, the present invention provides a method and system for delivering "on-demand" high fidelity music to computer systems via the Internet which does not involve proprietary audio players and/or encryption of the music. Additionally, the present invention provides a method and system that includes the above accomplishment and does not overly congest the communication networks of the Internet. Furthermore, the present invention provides a method and system that includes the above accomplishments and monitors the music (or media) delivered in order to compensate the owner of copyrighted music (or media) for it.

[**0088**] The foregoing descriptions of specific embodiments of the present invention have been presented for purposes of illustration and description. They are not intended to be exhaustive or to limit the invention to the precise forms disclosed, and obviously many modifications and variations are possible in light of the above teaching. The embodiments were chosen and described in order to best explain the principles of the invention and its practical application, to thereby enable others skilled in the art to best utilize the invention and various embodiments with various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention be defined by the Claims appended hereto and their equivalents.

What is claimed is:

1. A system for providing media content while preventing unauthorized access to a plurality of content sources, said system comprising:

a media content library having a plurality of media content pieces;

a global media content delivery network having said plurality of content sources, each content source of said plurality of content sources stores a portion of said media content library; and

a security system coupled to said plurality of content sources for allowing delivery of a media content piece from said media content library to an authorized client device while preventing unauthorized access to said media content library stored by said plurality of content sources.

2. The system as described in claim 1 wherein said global media content delivery network comprises a local area network (LAN), wide area network (WAN) or the Internet.

3. The system as described in claim 1 wherein said authorized client device is selected from a computer system, a desktop computer, a laptop computer, and a portable computing device.

4. The system as described in claim 1 wherein said media content piece comprises audio, music, video, a picture, graphics, or multimedia.

5. The system as described in claim 1 wherein said media content piece comprises a digital watermark or an embedded key.

6. The system as described in claim 1 wherein said security system for further providing an access key to said authorized client device for accessing said media content piece from a content source of said plurality of content sources.

7. The system as described in claim 6 wherein said access key is a time sensitive access key that becomes obsolete after a defined amount of time.

8. The system as described in claim 1 wherein said security system for further preventing access to said media content library when a media player application operating on a client device is determined invalid.

9. The system as described in claim 1 wherein said security system for further preventing access to said media content library when an identification provided by a client device is determined invalid.

10. A method for providing media content to a client device from a plurality of content sources while preventing unauthorized access to said media content, said method comprising:

transmitting a first request for a media content list of a media content library from said client device;

in response to receiving said first request, transmitting to said client device said media content play list together with a unique identification;

in response to receiving said media content list, transmitting a second request for delivery of said media content together with said unique identification from said client device;

provided said unique identification is valid, transmitting to said client device an access key together with a location of a content source of said plurality of content sources containing said media content;

in response to receiving said access key together with said location of said content source, transmitting a third request together with said access key to said content source from said client device; and

provided said access key is valid, transmitting said media content to said client device.

11. The method as described in claim 10 further comprising:

determining which content source of said plurality of content sources is nearest to the location of said client device.

12. The method as described in claim 10 further comprising:

determining which content source of said plurality of content sources contains said media content.

13. The method as described in claim 10 wherein said client device is selected from a computer system, a desktop computer, a laptop computer, and a portable computing device.

14. The method as described in claim 10 wherein said access key is a time sensitive access key that becomes obsolete after a defined amount of time.

15. The method as described in claim 10 wherein said media content is selected from an audio clip, a song, a video clip, a picture, a graphics picture, and a multimedia clip.

16. The method as described in claim 10 wherein said media content comprises a digital watermark or an embedded key.

17. The method as described in claim 10 wherein said location of said content providing source comprises an address to said media content.

18. The method as described in claim 17 wherein said address is changed after said transmitting said media content to said client device.

19. The method as described in claim 10 wherein each content source of said plurality of content sources stores a portion of said media content library.

20. The method as described in claim 10 wherein each content source of said plurality of content sources stores a copy of said media content library.

21. A system for providing media content while restricting unauthorized access to a plurality of content sources, said system comprising:

a media content library having a plurality of media content pieces;

a global media content delivery network having said plurality of content sources, each content source of said plurality of content sources stores a portion of said media content library; and

an authorization system coupled to said plurality of content sources for regulating delivery of a media content piece from said media content library to an authorized client computer while restricting unauthorized access to said media content library stored by said plurality of content sources.

22. The system as described in claim 21 wherein said global media content delivery network comprises a local area network (LAN), wide area network (WAN) or the Internet.

23. The system as described in claim 22 wherein said authorized client computer is selected from a computer system, a desktop computer, a laptop computer, and a portable computing device.

24. The system as described in claim 21 wherein said media content piece comprises audio, music, video, a picture, graphics, or multimedia.

25. The system as described in claim 24 wherein said media content piece comprises a digital watermark or an embedded key.

26. The system as described in claim 21 wherein said authorization system for further providing an access key to said authorized client computer for accessing said media content piece from a content source of said plurality of content sources.

27. The system as described in claim 26 wherein said access key is a time sensitive access key that becomes obsolete after a defined amount of time.

28. The system as described in claim 21 wherein said authorization system for further preventing access to said media content library when a media player application operating on a client computer is determined invalid.

29. The system as described in claim 21 wherein said authorization system for further preventing access to said media content library when an identification provided by a client computer is determined invalid.

* * * * *