US 20170026436A1

(19) **United States**

(12) **Patent Application Publication** (10) **Pub. No.: US 2017/0026436 A1**

Price (43) **Pub. Date: Jan. 26, 2017**

(54) **STREAMING MEDIA DELIVERY SYSTEM**

(71) Applicant: **WAG ACQUISITION, L.L.C.**, Flanders, NJ (US)

(72) Inventor: **Harold Edward Price**, Bethel Park, PA (US)

(21) Appl. No.: **15/283,581**

(22) Filed: **Oct. 3, 2016**

Related U.S. Application Data

(63) Continuation of application No. 13/815,040, filed on Jan. 25, 2013, which is a continuation of application No. 13/385,375, filed on Feb. 16, 2012, now Pat. No. 8,364,839, which is a continuation of application No. 12/800,177, filed on May 10, 2010, now Pat. No. 8,185,611, which is a continuation of application No. 10/893,814, filed on Jul. 19, 2004, now Pat. No. 7,716,358, which is a continuation-in-part of application No. 09/819,337, filed on Mar. 28, 2001, now Pat. No. 6,766,376.

(60) Provisional application No. 60/231,997, filed on Sep. 12, 2000.

Publication Classification

(51) **Int. Cl.**

H04L 29/06 (2006.01)

H04L 12/879 (2006.01)

(52) **U.S. Cl.**

CPC ***H04L 65/4076*** (2013.01); ***H04L 69/16*** (2013.01); ***H04L 49/901*** (2013.01)

(57) **ABSTRACT**

Streaming media, such as audio or video files, is sent via the Internet. The media are immediately played on a user's computer. Audio/video data is transmitted from the server under control of a transport mechanism. A server buffer is prefilled with a predetermined amount of the audio/video data. When the transport mechanism causes data to be sent to the user's computer, it is sent more rapidly than it is played out by the user system. The audio/video data in the user buffer accumulates; and interruptions in playback as well as temporary modem delays are avoided.

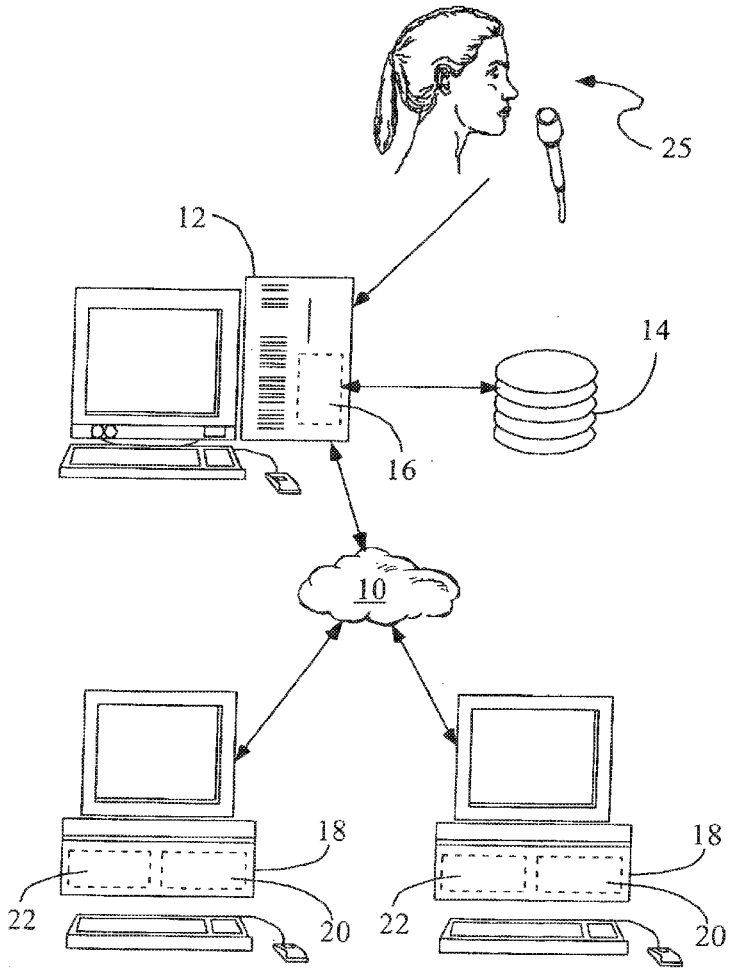


Fig. 1

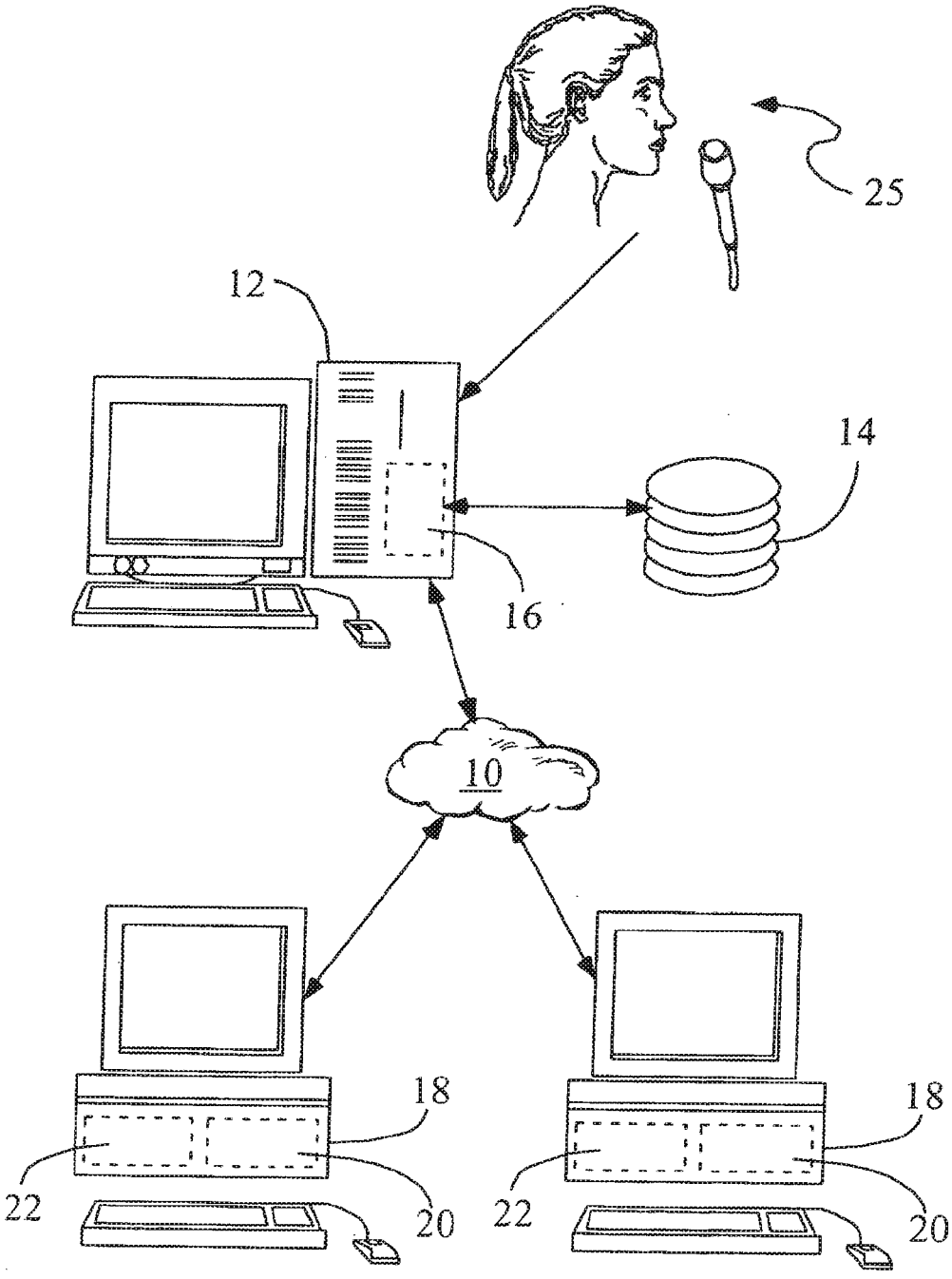


Fig. 2

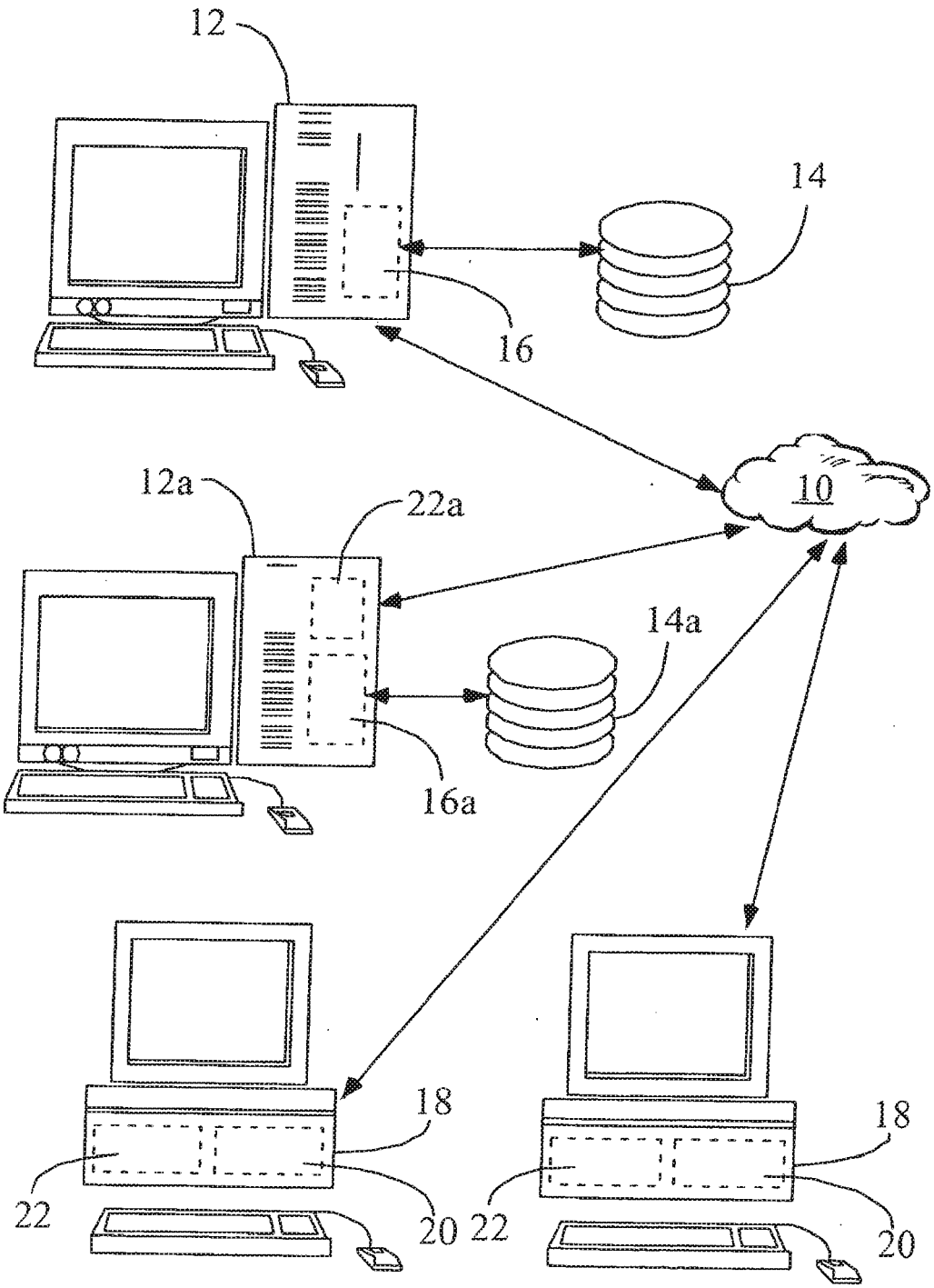
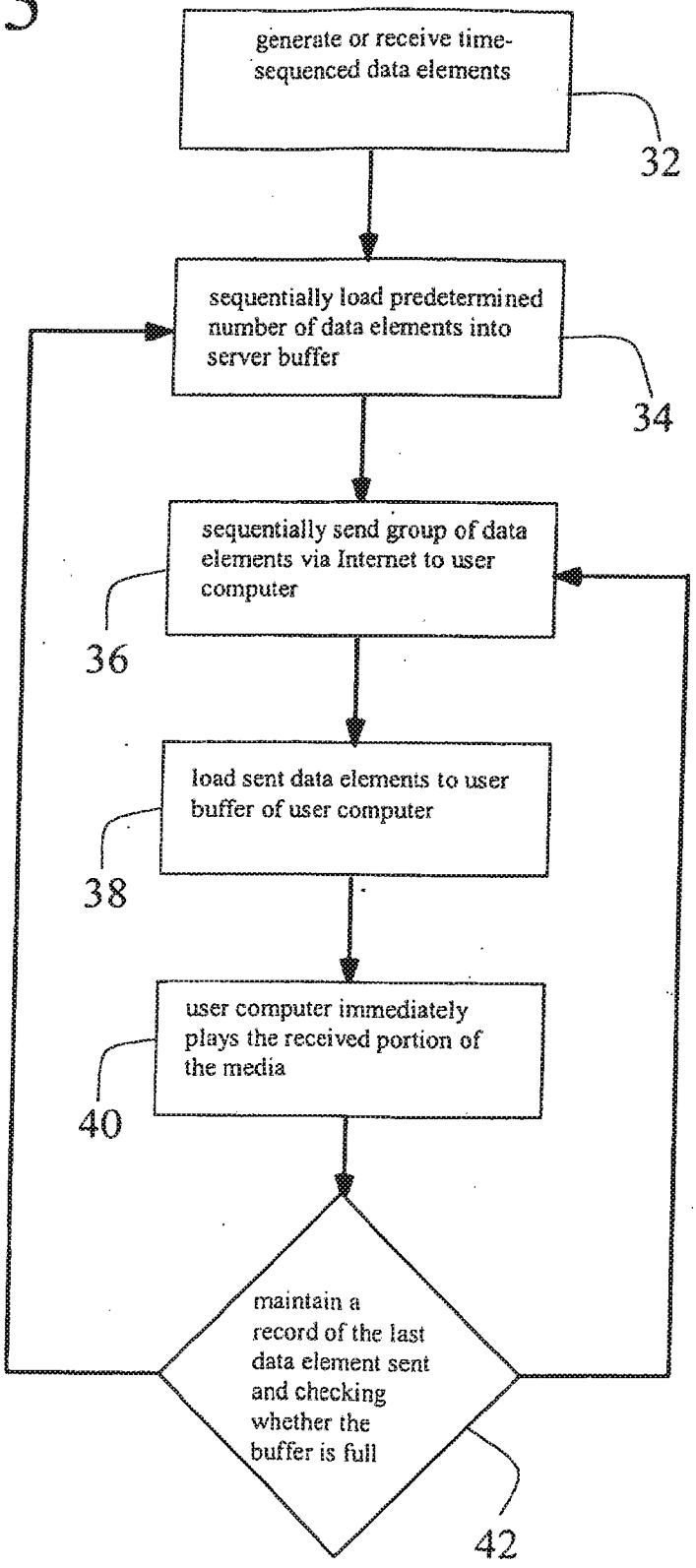


Fig. 3


generate or receive time-sequenced data elements
32
sequentially load predetermined number of data elements into server buffer
34
sequentially send group of data elements via Internet to user computer
36
load sent data elements to user buffer of user computer
38
user computer immediately plays the received portion of the media
40
maintain a record of the last data element sent and checking whether the buffer is full
42

STREAMING MEDIA DELIVERY SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] This application is a continuation of U.S. patent application Ser. No. 13/815,040, filed Jan. 25, 2013 (published on Jun. 13, 2013 as U.S. patent publication number 2013/0151724 A1), which was a continuation of U.S. patent application Ser. No. 13/385,375, filed Feb. 16, 2012 (published on Jun. 28, 2012 as U.S. patent publication number 2012/0166669 A1 and now U.S. Pat. No. 8,364,839, issued Jan. 29, 2013), which was a continuation of U.S. patent application Ser. No. 12/800,177, filed May 10, 2010 (published on Sep. 2, 2010 as U.S. patent publication number 2010/0223362 A1 and now U.S. Pat. No. 8,185,611, issued May 22, 2012), which was a continuation of U.S. patent application Ser. No. 10/893,814, filed Jul. 19, 2004 (published on Dec. 9, 2004 as U.S. patent publication number 2004/0249969 A1, and now U.S. Pat. No. 7,716,358, issued May 11, 2010), which was a continuation-in-part of U.S. patent application Ser. No. 09/819,337, filed Mar. 28, 2001 (now U.S. Pat. No. 6,766,376, issued Jul. 20, 2004), which was a nonprovisional of U.S. provisional patent application Ser. No. 60/231,997, filed Sep. 12, 2000 and now abandoned; and it claims the benefit, under 35 U.S.C. §120, of the respective filing dates of said nonprovisional applications, and the benefit under 35 U.S.C. §119(e) of said provisional application, as well as benefit under 35 U.S.C. §§120 and 119(e) (as applicable) of the filing dates of: copending U.S. patent application Ser. No. 10/825,869, filed Apr. 16, 2004 (published on Dec. 23, 2004 as U.S. patent publication number 2004/260828 A1), which was a continuation of said U.S. patent application Ser. No. 09/819,337, which was a nonprovisional of said provisional patent application Ser. No. 60/231,997; and hereby incorporates by reference the entire disclosure of each of said prior applications. This application further incorporates by reference the entire disclosure of U.S. patent application Ser. No. 13/374,942, filed Jan. 24, 2012 (published on Jun. 14, 2012 as U.S. patent publication number 2012/0151083 A1, and now U.S. Pat. No. 8,327,011, issued Dec. 4, 2012, which was a continuation of U.S. patent application Ser. No. 12/800,152, filed May 10, 2010 (published on Sep. 16, 2010 as U.S. patent publication number 2010/0235536 A1, and now U.S. Pat. No. 8,122,141, issued Feb. 21, 2012), which was also a continuation of said U.S. patent application Ser. No. 10/893,814.

BACKGROUND OF THE INVENTION

[0002] Field of the Invention

[0003] The present invention relates to multimedia computer communication systems; and more particularly, to systems and methods for delivering streaming media, such as audio and video, on the Internet.

[0004] Description of the Related Art

[0005] Prior to the development of Internet streaming media technologies, audio and video were formatted into files, which users needed to download in their entirety to their computers before the files could be heard or viewed. Real time, continuous media, as from a radio station, was not suitable for this arrangement, in that a file of finite size must be created so it could be downloaded. The advent of streaming media technologies allowed users to listen to or view the files as they were being downloaded, and allowed users to "tune-in" to a continuous media broadcast, or "stream", such as from a radio station.

[0006] Sending audio or video files via a network is known in the art. U.S. Pat. No. 6,029,194 to Tilt describes a media server for the distribution of audio/video over networks, in which retrieved media frames are transferred to a FIFO buffer. A clock rate for a local clock is adjusted according to the fullness of the buffer. The media frames from the buffer are sent in the form of data packets over the networks in response to interrupts generated by the local clock. In this manner, the timing for the media frames is controlled by the user to assure a continuous stream of video during editing. U.S. Pat. No. 6,014,706 to Cannon, et al. discloses an apparatus and method for displaying streamed digital video data on a client computer. The client computer is configured to receive the streamed digital video data from a server computer via a computer network.

[0007] The streamed digital video data is transmitted from the server computer to the client computer as a stream of video frames. U.S. Pat. No. 6,002,720, to Yurt, et al. discloses a system for distributing video and/or audio information, wherein digital signal processing is employed to achieve high rates of data compression. U.S. Pat. No. 5,923,655, to Veschi et al. discloses a system and method for communicating audio/video data in a packet-based computer network, wherein transmission of data packets through the computer network requires variable periods of transmission time. U.S. Pat. No. 5,922,048 to Emura discloses a video server apparatus having a stream control section that determines a keyframe readout interval and a keyframe playback interval, which satisfy a playback speed designated by a terminal apparatus. Finally, U.S. Pat. No. 6,014,694 to Aharoni, et al. discloses a system and method for adaptively transporting video over networks, including the Internet, wherein the available bandwidth varies with time.

[0008] Despite these developments, users viewing or listening to streaming content over Internet connections often encounter interruptions, due to the frequency of unanticipated transmission delays and losses that are inherent in many Internet protocols. These interruptions are commonly referred to as "dropouts", meaning that the data flow to the user has been interrupted (i.e., the audio "drops out").

[0009] Dropouts can be extremely annoying—for example, while listening to music. The current state-of-the-art solution to the problem uses a pre-buffering technique to store up enough audio or video data in the user's computer so that it can play the audio or video with a minimum of dropouts. This process requires the user to wait until enough of the media file is buffered in memory before listening or viewing can begin. The media data is delivered by a server computer, which has available to it the source of the media data, such as by a connection to a radio station. When the user connects to the server via the Internet, audio/video output at the user's system is delayed while the user's buffer is filled to a predetermined level. Typical pre-buffering wait times range from ten to twenty seconds or more, determined by the vendor providing the audio or video media. Even with this pre-buffering process, interruptions in playback still occur.

[0010] In this process, the user has a software application on the computer commonly called a "media player". Using the features built into the media player, the user starts the audio or video stream, typically by clicking on a "start"

button, and waits ten to twenty seconds or so before the material starts playing. During this time data is being received from the source and filling the media player's buffer. The audio or video data is delivered from the source at the rate it is to be played out. If, for example, the user is listening to an audio stream encoded to be played-out at 24,000 bits per second, the source sends the audio data at the rate of 24,000 bits per second. Provided that the user waits ten seconds, and the receipt of the buffering data has not been interrupted, there is enough media data stored in the buffer to play for ten seconds.

[0011] Gaps in the receipt of audio/video data, due to Internet slowdowns, cause the buffer to deplete. Because transmission of audio/video media data to the user takes place at the rate it is played out, the user's buffer level can never be increased or replenished while it is playing. Thus, gaps in the receipt of audio/video media data inexorably cause the buffer level to decrease from its initial level. In time, extended or repeated occurrences of these gaps empty the user's buffer. The audio/video material stops playing, and the buffer must be refilled to its original predetermined level before playing of the media resumes.

[0012] By way of illustration, if, in a ten second pre-buffering scenario, data reception stopped the instant that the media started playing, it would play for exactly ten seconds. Once the media data starts playing, it plays out of the buffer as new media data replenishes the buffer. The incoming data rate equals the rate at which the data is played out of the user's buffer, assuming the receipt of data across the Internet is unimpeded. If there are no interruptions in the receipt of the media data for the duration of the time the user listens to or watches the material, the buffer level remains constant and there will still be ten seconds of data stored in the media player's buffer when the user stops the player.

[0013] On the other hand, if the media player encounters interruptions totaling six seconds while playing the material, there would only be four seconds of media data remaining in the buffer when the user stopped it. If data reception interruptions at any time during the playing exceed ten seconds, the user's media player buffer becomes exhausted. There is no media data to play, and the audio or video stops—a dropout has occurred. At this point a software mechanism in the media player stops attempting to play any more of the material, and starts the buffering process again. The media player remains silent until the buffer refills, at which time the media player will once again start playing the material. This pattern has brought about considerable consumer frustration with streaming media over the Internet.

BRIEF SUMMARY OF THE INVENTION

[0014] There is a need for improved systems and methods for delivering streaming content over the Internet or other communications medium, which facilitate continuous transmission of streaming content, respond on demand without objectionable buffering delay, and perform without disruption or dropouts.

[0015] To address these objectives, various embodiments for delivering streaming content are provided, which envision that both the server and user systems involved in the content delivery may have buffering capacity. The embodiments make varying uses of this capacity to facilitate continuous content transmission on demand. Nearly instantaneous playback is achieved, while maintaining protection against playback interruption.

[0016] In one aspect, the server and user-sides of the transmission are coordinated, by (a) sending initial streaming media elements to the user system at a sending rate more rapid than the playback rate, to fill the user buffer; and (b) after the user buffer has been filled, sending further streaming media data elements to the user system at about the playback rate.

[0017] In another embodiment, the user system may be used to regulate transmission of streaming media to it, by a streaming media server. In such embodiment, the server may operate by (a) assigning identifiers to the sequential media data elements comprising the program; (b) receiving requests from the user system for media data elements corresponding to specified identifiers; and (c) sending media data elements to the user system responsive to said requests. A user system used in connection with such an embodiment may operate by (i) maintaining a record of the identifier of the last sequential media data element that has been received by said player; (ii) requesting transmission of the next sequential media data elements following said last sequential media data element, as said media player requires for continuous and uninterrupted playback.

[0018] Other aspects and advantages of the invention will be apparent from the accompanying drawings and the detailed description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

[0019] The invention will be more fully understood and further advantages will become apparent when reference is had to the following detailed description and the accompanying drawings, in which:

[0020] FIG. **1** is a schematic/block diagram illustrating the elements of a streaming media buffering system in accordance with one embodiment of the present invention;

[0021] FIG. **2** is a schematic/block diagram of an alternative embodiment of the system shown by FIG. **1**; and

[0022] FIG. **3** is a flowchart illustrating a method employed in one embodiment of the present invention.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

[0023] The following is a detailed description of certain embodiments of the invention chosen to provide illustrative examples of how it may preferably be implemented.

[0024] Audio and video media must play out over a period of time. Thus, in considering the delivery of such media, it is more appropriate in certain respects to think of bandwidth requirements than file size. The bandwidth requirement of audio or video media refers to the data rate in bits per second that must be transmitted and received in order to listen to or view the material uninterrupted.

[0025] Transmitting the audio or video material over a connection slower than the bandwidth requirement results in unsatisfactory viewing or listening, if viewing or listening is possible at all. The connection available may, for example, be by dialup modem, which has a maximum receive data rate of 56,000 bits per second. Audio and video encoded for distribution over the Internet may be compressed to be listenable or viewable within such a 56,000 bits per second bandwidth. Requirements for achieving adequate audio and video over the Internet may consume a considerable portion of the listener's available bandwidth.

[0026] There are two types of encoding schemes used for audio and video material—"Variable Bit Rate" (VBR), and "Constant Bit Rate" (CBR). CBR encoding represents the encoded media with a constant bit rate per second, regardless of the complexity of the material being encoded. For example, if an audio source is encoded at 20 kilobits per second at a Constant Bit Rate, the media data being produced from the encoding is at 20 kilobits per second, whether the audio material is complex (e.g., symphonic) or silence. Variable Bit Rate encoding uses a variable number of bits to represent sounds or video, with more bits required for complex material (e.g., symphonic sounds or action scenes) than for simple sounds, silence, or still scenes. The most usual encoding scheme used for streaming media is CBR, because the resulting data rate is more predictable than for VBR. Statements in this specification concerning "constant" data rates and the like should be understood as subject to appropriate variation where VBR-encoded data may be involved.

[0027] Even if a user's Internet connection has the requisite average bandwidth capacity to allow reception of the program, the actual rate of delivery of data to the user can fluctuate widely above, and more particularly, below, this average, as a function of the quality of the user's connectivity at any given time. Internet connection quality can vary rapidly over time, with two primary factors responsible for degradation of the instantaneous bandwidth actually available to the user. These factors are the quality of the user's Internet connection, which can have periods of interference causing reduced available bandwidth, and momentary Internet congestion at various points along the route over which the user's data flows. Each of these factors can cause delays and interruptions in the transmission of data to the user. Internet data communications devices such as routers are designed to drop data packets if they get overloaded. For material that is not time sensitive, these dropped packets will usually be resent, and the user will eventually be presented with the material. However, since streaming media is time sensitive, dropped packets can have a significant impact on the receipt and playback of an audio or video stream. Such degradation in the receipt of Internet data is very common, and prevent most users from being able to listen to or view streaming media without interruption unless some special provisions have been incorporated into the user's computer software to accommodate data transmission interruptions.

[0028] There are two fundamental types of streaming media, which affect, in some respects, the requirements for smooth and continuous delivery: (i) material that originates from a source having a realtime nature, such as a radio or TV broadcast, and (ii) material that originates from a non-real-time source such as from a disk file. An example of non-real-time material might be a piece of music stored as a disk file, or a portion of a broadcast that originally was realtime, perhaps yesterday's TV evening news, and was recorded into a disk file. For purposes of clarity within this document, streaming media of type (i) will be referred to as "real time" or "broadcast" media, and streaming media of type (ii) will be referred to as "file based" media.

[0029] In many respects, both streaming media types are handled similarly in conventional systems, and both are handled similarly (in a number of respects) by the streaming media delivery system of the present invention. Nevertheless, the two streaming media types are readily distinguished. Broadcast streaming media has as its source a system or arrangement that by definition can only be transmitted to users as fast as the material is generated; for example, a disk jockey speaking into a microphone. File based media, on the other hand, can be transmitted to users at any available data rate, since in the context of data communications, the time required for reading a small portion of data from a file residing entirely on a locally accessible, random access storage device may be considered negligible.

[0030] In conventional systems for streaming media over the Internet, media data (whether real-time or file based) is simply transmitted from the server to the user at the rate at which it will be played out (the "playback rate"), regardless of the data rate capabilities of the connection between the server and the user.

[0031] Conventional streaming media systems may incorporate server-side buffering systems for programmatic purposes. For example, the system may buffer media data at the server for the purpose of packet assembly/disassembly. Media data may also be buffered at the server to permit programming conveniences such as dealing with blocks of data of a specific size. However, conventional streaming media systems have not utilized server-side buffering for the purpose of mitigating long term Internet performance degradation. Rather, prior art systems, in which data is continuously transmitted at the playback rate, have performed buffering for continuity purposes solely on the user side, with the consequences discussed above of startup delays and dropouts. The present invention addresses such shortcomings.

[0032] The present invention provides a system and method for delivering streaming media, such as audio or video media, via the Internet or other communications medium. Immediate playing of the media on a user's computer is afforded, while reducing interruptions in playback due to Internet congestion, and temporary modem delays due to noisy lines. Nearly instantaneous playback is achieved, while maintaining protection against playback interruption. Delayed starts, heretofore required to provide protection against interruption, are avoided. Data lost due to interruptions in the receipt of media data by the media player can be recovered while the player continues to play out the audio or video material. If the interruptions are so severe as to deplete the user's buffer and stop the play out, the media player can quickly recover as well, by beginning to play out again without waiting to first build up the buffer, as soon as the media player begins to receive media data elements.

[0033] In one embodiment, the invention provides a system for distributing via the Internet streaming media composed of a plurality of time-sequenced data elements. As shown in FIG. **1**, the system is provided with a server **12** connected to the Internet **10** for transmitting the streaming media data elements. Associated with the server **12** is a server buffer **14** for storing at least one of the data elements for transmission, and a buffer manager **16**. Buffer **14** is a conventional computer storage mechanism such as a hard disk, as shown for convenience of illustration, or, preferably, an electronic storage arrangement such as Random Access Memory (RAM).

[0034] The media may come from a live source, shown as **26** in FIG. **1**, or from a stored file on the server **12**, or another storage device, such as a hard drive.

[0035] A number of different implementations of such a server, involving different ways of handling server buffer **14**, will be discussed.

[0036] In the various implementations, there is in each case at least one user computer **18** (or similar device) connected to the server **12** via the Internet **10** or other data communications medium. User computer **18** is associated with media player software incorporating user buffer **20**. The user buffer **20** is provided with means for storing a predetermined number of the data elements. User buffer **20** is a conventional computer storage mechanism such as a hard disk, or, preferably, an electronic storage arrangement such as Random Access Memory (RAM) as suggested by the illustration. A buffer manager **22** is also associated with the user computer **18**. The buffer manager **22**, having the form of software or firmware, is provided with means for receiving and storing a predetermined number of media data elements which are received sequentially by the media player, playing the data out sequentially as audio and/or video, and deleting media data elements from the buffer as they are played out (or displacing them by newly arrived elements). As data is played out, the next sequential data elements are received from the server in such a fashion as to approximately maintain the predetermined number of data elements in the user's buffer. It should be understood that data might arrive at the media player out-of-sequence and that processes in the media player or the media player buffer manager are responsible for properly arranging this data.

[0037] Alternatively, user computer **18** may be replaced by an Internet radio or Internet Appliance, which is comprised of a dedicated processor for receiving Internet radio or audio/video material. Examples of such devices might range from familiar computing devices such as palmtops, PDAs (Personal Digital Assistants), and wireless phones, to devices that appear and operate similarly to conventional consumer electronic devices such as radios and televisions, but with the additional capability of Internet access.

FIFO Server Buffer Implementation

[0038] There are a large number of ways of managing server buffer **14** in order to implement the systems and methods described in this specification. In one implementation, buffer manager **16** is adapted to effectively render server buffer **14** a FIFO device. In this implementation, buffer manager **16** is provided in the form of software or firmware that provides means for: receiving the media data; supplying media data in order to the FIFO buffer; supplying the buffer **14** with a predetermined number of data elements; maintaining pointers **24***a* through **24***n* into the buffer, one for each user computer indicating the last media data element that has been sent to that user, thus indicating the next element or elements to be sent; and, once the FIFO buffer is full, deleting (or displacing) the oldest data element in the buffer as each new data element is received. These means are arranged to maintain the pre-determined number of data elements in the FIFO buffer. Buffer Manager **16** may also comprise means for digitizing, encoding, and packetizing the media data, and formatting media data according to the requirements of buffer **14**.

Data Window Buffer Implementation

[0039] If the media source is file based, such as a music clip stored as a disk file, and if the disk file is stored on the server or an associated server computer, the server's connection to the source could be considered to be near instantaneous. In this case, rather than audio/video data filling and depleting the buffer **14**, an amount of audio/video data equivalent to the desired buffer size may be logically constituted as a FIFO buffer. Such a construct is commonly called a data window. The data window moves on a time-sequenced basis through the media data file, thus defining the contents of the buffer on a moment-by-moment basis and performing the equivalent functions to receiving a new data element and deleting the oldest data element.

Example Buffering Methods

[0040] In an arrangement that receives media data directly or indirectly from a real-time source, such as a radio station, server buffer **14** might be set to hold (for example) 30 seconds of media data. Because the source produces media data in real time, the media data is delivered to the server approximately at the rate it is generated.

[0041] Of course, there can be variability in this data delivery process due to networking, disk accesses, and so on, causing the delivery rate of the media data to be variable over short periods of time, typically measured in seconds. But over a longer period of time measured in minutes or tens of minutes or longer, the media data is delivered from source to server at the rate it is generated, and the server in turn provides that media data to the FIFO buffer at that same rate. Since CBR encoding is typically used for streaming media, the media data is generated, received by the server, and provided to the buffer approximately at a fixed rate.

[0042] The server buffer **14** is filled the first time the media source connection is established or a disk file is read. The amount is preferably adequate to bridge gaps typical of Internet and modem delays to the user. This buffer may, for example, hold enough data elements for about one minute of play.

[0043] Once server buffer **14** is full, for each new data element received into the buffer the oldest data element is deleted (or displaced) from the buffer. In some implementations, requests from user computers to connect may not be accepted until server buffer **14** is full.

[0044] Once a connection is made to a user's computer (e.g., user computer **18**), server **12** sends the media data to the user computer in the following manner. First, media data is sent to the user computer at a rate faster than the playback rate, which may be the highest rate that the data connection between the server and the user computer will support, or any lower rate that is a higher rate than the playback rate (referred to herein as a "higher than playback" rate), until the predetermined amount of data that had been stored in the server buffer has been transferred to the user's computer. Once the contents of server buffer **14** has been transferred, a steady state condition is reached wherein as each media data element arrives at server **12**, it is immediately sent out to the user computer. In this steady state condition, the media data is sent at a rate that matches the constant fill rate of the server buffer, and is received at the same rate by the user computer if there are no interruptions in the transmission of media data between the server and the user's computer (with some variation in the case of VBR content). If interruptions have interfered with the arrival of sent media data to the user's computer, that data may have been "dropped" by routers in the Internet and needs to be resent. This causes data to "back up" into the server FIFO for that user.

[0045] A data communications transport mechanism, such as the TCP protocol, may be used for the reliable delivery of data in an ordered sequence from the source of the media data to the server, or from the server to the media player software of the user computer. Resending missing data is the responsibility of the reliable transport mechanism. The server buffer **14** "sends" data by delivering it to the transport mechanism. The transport mechanism actually manages transmission of the data across the communications medium, and has processes to determine if all the data that has been sent has been received by the destination. If not, missing pieces of data are automatically resent to the destination, and are arranged to be delivered to the target software on the destination system in an ordered fashion. In this example, the destination is user computer **18**, and the target software on the destination system is the media player. If the transport mechanism determines that data is missing, it retransmits that data to the destination at a higher than playback rate. In another method of operation, server **12** can use an unreliable transport mechanism, such as UDP, and rely on a streaming software process to manage data delivery and the resending of data elements not received by the media player.

[0046] All media data to be delivered to a user computer may be sent at a higher than playback rate, either by the server buffer **14** passing media data to the transport mechanism, or by the transport mechanism delivering or redelivering the media data to the user computer.

[0047] This is enabled by buffering data at the server **12**, and is distinctly different from prior art, in which media data is only sent from the server **12** to the user computer **18** at the rate at which it is to be played out.

[0048] As an example of the preceding description, if the server had been set to store 30 seconds of audio in its buffer, when a user connects, that 30 seconds worth of media data is transferred to the user's media player buffer at a higher than playback rate. The media player can begin playing as soon as it has received a very minimum amount of data, perhaps comprising only a single packet of media data.

[0049] For ease of understanding, consider the server buffer and the media player buffer to be an elastic system that between the two stores (for example) up 30 seconds of audio data. The server starts with 30 seconds of buffered audio data which it transfers to the media player until the server has no buffered media data and the media player has 30 seconds of buffered media data. Regardless of how much of the buffered media data has been transmitted to the media player, there always is 30 seconds of media data being buffered between the two locations. Consequently, the audio being played out by the media player will always be 30 seconds behind the audio at the source. If there were a media player in the radio station studio, an announcer would hear themselves through the media player with a 30 second delay.

[0050] Connections from the server **12** through the Internet **10** commonly are much faster than the data rate required for audio or video playback. This fact is insignificant for conventional servers because, not having a FIFO buffer or a buffer pointer for each user, audio/video data can only be sent as fast as it becomes available, or as fast as the pace at which it must be delivered to the user in order to be properly replayed. The user, typically interacting with media player software on the user's computer, selects a media source requiring a data rate slower than that available by the user's connection to the Internet. For example, if the user's connection to the Internet is made via a 56,000 bits per second modem, the user might select a media source encoded for playback at 24,000 bits per second.

[0051] With the present invention, as soon as a user connects to the server **12**, the server **12** transmits audio/video data as sequential data elements from its buffer **14** to the buffer **20** of the user, at a higher than playback rate. Unlike the prior art, media begins to play on the user computer **18** as soon as the user connection is made to the audio server **12** and a minimal amount of data elements have been received and stored in the user's buffer **20**. The user's buffer **20** is built up while the media is playing. As each data element is played, it is deleted or displaced from the user's buffer **20**.

[0052] Initially, the user buffer manager **22** requests the server **12** to send media data elements to start the playback stream, such as by selecting a radio station from a list. The server **12** responds by sending data elements to the user computer **18** at higher than the playback rate, until the entire FIFO buffer **14** has been sent to the user computer. Upon receipt of the initial data elements, the user buffer manager **22** begins playback. Because (with reference to CBR content) this is a synchronous system with the source, server, and user computer operating by the same playback clock rate as determined by the encoding rate of the media, as each data element is played out and is deleted or displaced from the user buffer **20**, another data element has been deposited into the server buffer **14** and is available to be sent to the user computer. Server **12** sends the newly available data elements at a higher than playback rate.

[0053] Since the connection from the Internet to the user is faster than that required for media playback, audio/video data is transmitted from the server faster than it is played out by the user system, thus building up audio/video data in the user buffer. For example, if the user's connection to the Internet is at 56,000 bits per second, and the data rate encoded for the media to be played is 24,000 bits per second, the buffer level of the user buffer **20** will fill at the rate of 32,000 bits per second (56,000 bits per second receive rate, minus 24,000 bits per second playout depletion rate).

[0054] If, for example, the server buffer **14** held one minute of audio/video data, eventually the user buffer **20** will hold one minute of audio/video data. The effect is that, over a brief period of time, the server buffer **14**, or a designated portion of it, is transferred to the user buffer **20**. In one embodiment, the number of data elements in the server buffer **14** actually never changes, it always maintains one minute of audio/video data. However, for the particular user under discussion, a copy of all the data held in the buffer has been sent to the user. Since the user buffer **20** now holds one minute of audio/video data, it can play continuously despite data reception interruptions of less than a minute.

[0055] Where some media data has been resent by the reliable transport layer, there may be more data to be sent than would be sent at the routine constant fill rate, and in such a case the server transport mechanism will again send the buffered media data at higher than the playback rate. Similarly, if the media player buffer begins to deplete or becomes depleted due to networking interruptions, the server will attempt to send as much data as is necessary to rebuild the user computer's buffer to the proper level, again at higher than a playback rate. This allows for rebuilding the user's computer buffer under circumstances wherein Internet interruptions have blocked the normal flow of data.

[0056] Thus, as soon as the interruption ceases, the user buffer **20** can begin to rebuild, which will take place at higher than the playback rate. The media player can continue to play out the audio/video material while the user buffer **20** rebuilds. When compared to conventional systems, which provide no capability to rebuild the user's computer buffer when data is lost, the streaming media buffering system of the present invention provides for recovery of lost data elements and the restoration of the user's buffer, even while the user media player continues to play.

[0057] Under conditions in which interruptions have interfered with the arrival of sent media data to the user's computer, data loss exceeding certain levels will cause the transport mechanism software to stop accepting data for transmission from the application software, namely the streaming media server software. Although other arrangements are possible within the scope of this invention, in preferred embodiments, the streaming media server software keeps track of the last data element in the FIFO buffer that has been sent to each user, using a software pointer. Alternatively, or in addition, a feedback manager may be associated with user computer **18**, including means for sending to the source server the serial number of the last data element received, or for requesting more data. An interruption in the ability to send media data to a user results in the "last element" pointer "backing up" in FIFO buffer **14** in such a way that the server knows from what point in the buffer to restart sending data when the transport mechanism again requests data to send. When the server software receives that notification, it will begin sending data to the user starting from the next data element to send as indicated by the pointer, and sending as much data as the transport mechanism will accept. The transport mechanism will again send this data as fast as it can to the user. This process continues until the steady state condition is again reached wherein each data element is sent to the user as soon as it arrives from the media source, and a pre-determined number of data elements are maintained in user buffer **20**.

[0058] The predetermined buffer level in the user buffer **20** may be set at less than the predetermined buffer level of the server buffer **14** if desired. For example, the server buffer **14** might be set to hold one minute of media data, and the user buffer **20** might be set to hold thirty seconds of media data. In another embodiment, a feedback manager **62** is associated with the user computer **18**. The feedback manager **62** is provided with means for sending to the source server **12** the serial number of the last data element received. Feedback manager **62** has the form of software or firmware that tracks the last data element received and loaded into the user buffer. In addition, feedback manager **62** is adapted to send the serial number to the source server **12**. In this manner, the source server **12** sends the media as sequential data elements at a rate dependent on the quality of the connection with each user computer **18**. The media may come from a live source, shown as **25** in FIG. **1**, or from a stored file on the source server **12**, or another storage device, such as a hard drive.

Implementation with Feedback Manager

[0059] The buffer manager at the source server effectively renders the source buffer **14** a FIFO device holding a fixed amount of data with a constant, time-sequenced fill rate and a constant, time-sequenced depletion rate. Each audio/video data element carries a sequential serial number. Once the buffer **14** is full, each new audio/video data element, identified by a higher serial number, displaces the oldest audio/video data element, identified by the lowest serial number in the buffer **14**. In the case of an instantaneous media source, rather than audio/video data filling and depleting the buffer, the top and bottom pointers spanning an amount of audio/video data equivalent to the desired buffer duration move synchronously on a time-sequenced basis to the next higher serial number of the audio/video data available in the system, thus defining the contents of the buffer on a moment-by-moment basis. Thus, if the buffer is capable of holding 100 audio/video data elements, constituting one minute of audio playback, the audio/video data elements within the buffer would hold serial numbers of B (baseline)+Tr (transmitted)+(0-99), wherein, starting at some arbitrary value B for the baseline, the serial number count would have been incremented by the number Tr representing the total number of data elements that have been transmitted, and the buffer at any point in time would hold the audio/video data elements in the range B+Tr+O to B+Tr+99. On the next clock tick, the buffer holds B+(Tr+I)+O to B+(Tr+I)+99.

[0060] The unique pointer assigned to each user identifies by serial number either the last data element that was sent to that user, or the next data element to be sent. The selection of either mode is arbitrary; but whichever mode has been selected, that mode is systematically implemented. For purposes of this document, we will use the "last data element that was sent." Thus, for any user, the pointer represents the serial number B+Tr+x, where x represents some value between 0 and 99, as being the serial number of the last audio/video data element that had been sent to the user. Each time a data element is transmitted to the user, x is incremented, pointing to the next higher value in the buffer. Each time a new data element is deposited in the buffer by the audio source, x is decremented. Since audio/video data elements are transmitted to the user faster than they are deposited into the buffer, x will increment faster than it decrements and over time will equal the maximum value of 99, pointing to the most recently deposited audio/video data element.

[0061] The amount of data stored in the source server buffer **14** remains the same, regardless of the pointer value associated with any individual user. The pointer indicates the last data element that has been transmitted to the user, and thus also identifies the next data element to be transmitted to the user.

[0062] When the user's pointer equals B+Tr+99, which is the most recently deposited audio/video data element, the user computer **18** receives audio/video data in real time from the media source. The moment the next audio/video data element is deposited into the source server buffer **14**, a copy of that data element is transmitted to the user. The user buffer **18** will now contain 100 audio/video data elements, representing one minute of audio/video data, that will be played on a FIFO basis. In effect, the source server buffer **14** has been moved to the user buffer **20**.

[0063] Since the user buffer **20** now holds one minute of audio/video data, it can play continuously despite data reception interruptions of less than a minute, and as soon as the interruption ceases the user buffer **20** can begin to rebuild.

[0064] The user computer **18** and the media server **12** are synchronized by a feedback manager in which the user computer **18** either acknowledges the receipt of the serialized audio/video data packets, or requests the next increment

of audio/video data packets. This feedback enables the source server to keep track of the buffer pointer on each user's system.

[0065] Interrupts or delays in the flow of data from the source server to the user will cause the user's system to play audio out of the buffer without the buffer being replenished at the same rate. Consequently, the user buffer pointer will decrement at a faster pace than it increments, and the feedback mechanism will keep the source server buffer pointer for that user synchronized. All of the users' source server buffer pointers also decrement with each tick of the clock, as data flows in and flows out of the fixed size buffer. Thus, as a user buffer drains down, the user's source server buffer pointer indexes down as well, in lockstep. Since new audio/video data is continuously placed into the source server buffer from the source, this has the effect of rebuilding the user's buffer at the server.

[0066] Once this system is set in motion, a buffer of a preset duration is constantly maintained for each user, partially or completely at either source server or the listener's system, or ebbing and flowing between them as a result of moment-to-moment circuit conditions.

Distribution Fed from a Separate Source

[0067] In another embodiment, the buffer concept of this invention can be daisy-chained between multiple Servers. For example, a system might include a source server computer co-located in a radio station studio, which transmits to a network distribution server resident in a data center, to which users would connect. The source server would fill its buffer, transfer the buffer to the network distribution server using the process just described for transferring a buffer from a source server to a user, and then the network distribution server would transfer its buffer to the user, again, using the process just described except now with the network distribution server replacing the source server in delivering audio/video data to the user system.

[0068] Such an embodiment is shown in FIG. **2**. In this embodiment, the media source may be separate from the server **12**, such as computer system **28** located at a broadcast media source, such as a radio station studio. Computer system **28** is a logical element in a data network, and can be physically collocated with the audio source, such as a computer resident in a radio station studio, or it can be remote from the audio source, such as a computer in a data center receiving digitized audio from a distant radio station.

[0069] This computer system **28** includes a source manager **30** which may be implemented in software or firmware. The source manager **30** comprises means for: receiving media data elements as they are generated by the audio and/or video source, formatting media data according to the requirements of server **12**, buffer **14**, and buffer manager **16**; and, for transmitting that media data to server **12** as they are generated. Source manager **30** may, as part of such formatting, include means for digitizing, encoding, and packetizing the media data. Media data typically is generated in real time such as by a speaker talking into a microphone or by playing a CD.

[0070] Generally, computer system **28** transmits media data to server **12** in real time as the media data is generated. Buffering of media data might occur at computer system **28** for convenience of programming, but such buffering is incidental to the operation of the end-to-end system being described. Computer system **28** connects via the Internet **10**, or other suitable data communications medium, to a server **12**, wherein server buffer manager **16** receives the media data for input into the FIFO buffer **14** as described previously, and maintains the pre-determined number of data elements in the FIFO buffer.

[0071] Server **12**, in turn, transmits the media data to one or more user computers **18**, also as previously described.

Example Methods

[0072] In another embodiment, shown in FIG. **3**, the invention provides a method for distributing from a server via the Internet streaming media composed of a plurality of time-sequenced data elements.

[0073] Time-sequenced data elements are generated or received **32**. Next, a predetermined number of the data elements is sequentially loaded **34** into a server buffer, which process of **32** and **34** continues indefinitely as long as there is media data available. Next, a group of the data elements is sequentially sent **36** via the Internet from the server buffer to a user computer connected to the Internet, more rapidly than they are played out by the user system. Upon receipt by the user computer, the sent group of data elements is loaded **38** into a user buffer associated with the user computer. The user computer immediately plays **40** the received portion of the media on the user computer. At **42**, if the user buffer is not full, then additional data elements are sent to the user computer **36**, again more rapidly than it is played out by the user system. And also at **42**, if the user buffer is full, the system waits until new media data is delivered to the server buffer **34**. This process is repeated until the entire media file is played at the user computer.

[0074] In another embodiments, the steps depicted in FIG. **3** could be modified as follows. A serial number is assigned **30** to each of the plurality of time-sequenced data elements. Next, a predetermined number of the data elements is sequentially loaded **32** into a source buffer, and a group of the data elements is sequentially sent **34** via the Internet from the source buffer to a user computer connected to the Internet. Upon receipt by the user computer, the sent group of data elements is loaded **36** into a user buffer associated with the user computer. Then, the user computer sends **38** to the source server the serial number of the last data element received by the user computer. The user computer immediately plays **40** the received portion of the media on the user computer. This process is repeated until the entire media file is played at the user computer. Unlike conventional buffer arrangements, audio begins to play on the user system as soon as the user connection is made to the audio source server. The user's buffer is built up while the audio is playing. Advantageously, the system and method of this invention create a faster than real time connection. That is to say, audio/video data is transmitted from the server faster than it is played out by the user system, thus building up audio/video data in the user buffer.

[0075] In another embodiment, the server is connected to the Internet and provisioned as initially described, and has available to it file based media data as the source material. The file based media data can be read by the server which can deliver media data elements to the server FIFO buffer to the same effect as if the data had arrived from a broadcast media source. As before, the server provides a buffer manager and a FIFO buffer, and provides a means for receiving the sequentially arranged media data elements from the file based media source and storing those data elements in the FIFO buffer.

[0076] The buffer manager comprises means for: receiving the media data; supplying media data in order to the FIFO buffer; supplying the FIFO buffer with a predetermined number of data elements; maintaining a pointer into the buffer for each user computer indicating the last media data element that has been sent to that user, thus indicating the next element or elements to be sent; and, once the FIFO buffer is full, deleting the oldest data element in the buffer as each new data element is received, said means arranged to maintain the predetermined number of data elements in the FIFO buffer. The server buffer manager, or a separate process on the server, or a process on another computer having access to the file based media data, provides for reading the media data file and making available to the FIFO buffer sequentially arranged media data elements. At least one user computer is connected to the server via the Internet. The user computer is associated with a media player software incorporating a user buffer and comprises means for receiving and storing a predetermined number of media data elements which are received sequentially by the media player, playing the data out sequentially as audio and/or video, and deleting media data elements from the buffer as they are played out. As data is played out, the next sequential data elements are received from the server in such a fashion as to approximately maintain the predetermined number of data elements in the user's buffer.

[0077] In another embodiment, the server is connected to the Internet and provisioned as initially described. The server buffer manager, or the media source, provides for sequentially numbering the media data elements. The server buffer manager does not maintain a pointer into the server buffer for each user. Instead, the media player buffer manager in the user computer maintains a record of the serial number of the last data element that has been received.

[0078] Via the use of standard data communications protocol techniques such as TCP, the user computer transmits a request to the server to send one or more data elements, specifying the serial numbers of the data elements. The server responds by sending the requested data elements, and depends upon the reliable transmission protocol to assure delivery. The user computer then continues with additional data requests for the duration of playing the audio/video material. In this manner, the user computer, not the server, maintains the record of the highest data element number stored in the user computer buffer. The media data will be transmitted to the user computer as fast as the data connection between the user computer and the server will allow. As before, the server provides a buffer manager and a FIFO buffer, and provides a means for receiving the sequentially numbered media data elements from a broadcast media source or a file based media source, and storing those data elements in the FIFO buffer. The buffer manager comprises means for: receiving the media data; supplying media data in order to the FIFO buffer; supplying the FIFO buffer with a predetermined number of data elements; and, once the FIFO buffer is full, deleting the oldest data element in the buffer as each new data element is received.

[0079] Such means is arranged to maintain the pre-determined number of data elements in the FIFO buffer. At least one user computer is connected to the server via the Internet.

[0080] The user computer is associated with a media player software incorporating a user buffer and comprises means for receiving and storing a predetermined number of media data elements which are received sequentially by the media player, playing the data out sequentially as audio and/or video, and deleting media data elements from the buffer as they are played out. As data is played out, the next sequential data elements are requested from the server in such a fashion as to approximately maintain the predetermined number of data elements in the user's buffer.

[0081] In yet another embodiment, the invention provides a method for distributing from a server via the Internet streaming media composed of a plurality of time-sequenced data elements. A predetermined number of the data elements are sequentially loaded into a FIFO buffer. Additional data elements continue to be received.

[0082] As each new data element is input to the buffer, the oldest data element is deleted from the buffer, maintaining in the buffer the same predetermined number of data elements. At the request of a user computer for connection to a media stream, a group of the data elements is sequentially sent via the Internet from the FIFO buffer to the user computer connected to the Internet. Upon being received by the user computer, the sent group of data elements is loaded into a user's buffer associated with the user computer.

[0083] The user's computer immediately begins to play the audio/video streaming media material. The server continues to send the next data elements in sequence until the contents of the FIFO buffer have been sent. The data elements are sent by the server as fast as the connection between the server and user computer will allow. Once the contents of the FIFO buffer have been sent to a user computer, as each new data element is received into the FIFO buffer it is immediately sent to the user computer in such a manner as to keep the user computer buffer full. The process repeats for substantially the entire time that the audio/video material is played.

[0084] Unlike conventional buffering systems, audio begins to play on the user system as soon as the user connection to the audio server is effected and a small amount of data has been transferred-conventional systems required many seconds of data. Audio/video media data is initially transmitted from the server more rapidly than it is played out by the user system, until the server buffer has been transferred to the user computer. The user's buffer is built up while the audio is playing, and can be restored if it is diminished by data transmission interruptions. Advantageously, the system and method of this invention afford faster data transmissions than the playback data rate of the media data. Audio/video data is transmitted from the server more rapidly than it is played out by the user system under conditions wherein the user's computer buffer is not full.

[0085] The audio/video data in the user buffer accumulates; interruptions in playback due to temporary Internet and modem delays are avoided. It should be realized that, although the invention has been described hereinabove in connection with a process wherein the server sends buffered media data to the user "as fast as the network connection will permit", it is adequate, as mentioned in this paragraph, that the buffered data be transferred from the server to the user at a rate faster than the playback rate.

[0086] Although the preferred embodiment utilizes a reliable transport mechanism to move data between the server and the user, alternative embodiments could incorporate this invention's buffering system in combination with an unreliable datagram-based transport mechanism.

[0087] Thus, it can be seen that the present invention provides a system and method for sending streaming media,

such as audio or video files, via the Internet. Immediate playing of the media on a user's computer is afforded while reducing interruptions in playback due to Internet congestion and temporary delays. Delayed starts, heretofore required to provide protection against interruption, are avoided. Data loss due to interruptions in the receipt of media data by the media player can be recovered while the player continues to play out the audio or video material. If the interruptions are so severe as to deplete the user's buffer and stop the play out, the media player will begin to play out again as soon as the media player begins to receive media data without waiting to first build up the buffer.

[0088] Having thus described the invention in detail, it should be understood that various changes, substitutions, and alterations may be readily ascertainable by those skilled in the art, and may be made herein without departing from the spirit and scope of the invention as defined by the claims.

I claim:

1. A method for operating a media player to receive and play an audio or video program, from a remote media source via a data connection over the Internet, the method comprising:

sending requests from the media player to the media source via the data connection for one or more serially identified media data elements representing the audio or video program, each media data element comprising a digitally encoded portion of the audio or video program and having a playback rate, each request specifying one or more serial identifiers of the media data elements requested;

receiving each of the requested media data elements via the data connection, wherein the data connection has a data rate more rapid than the playback rate of the media data elements, and each received media data element is received at a rate as fast as the data connection between the media source and the media player allows;

storing the received media data elements in a memory of the media player;

playing the received media data elements in series from the memory of the media player; and

as the received media data elements are played, sending additional requests for subsequent media data elements for storage in the memory of the media player as required to maintain about a predetermined number of media data elements in the memory of the media player during playing.

2. The method of claim **1**, further comprising maintaining in the memory of the media player a record identifying the last media data element received and stored by the media player.

3. The method of claim **1** wherein the serial identifiers are sequential.

4. The method of claim **1**, wherein the receiving is via a reliable transmission protocol.

5. The method of claim **4**, wherein the reliable transmission protocol is TCP.

6. A device comprising a media player operable to receive and play an audio or video program, from a remote media source via a data connection over the Internet, the media player comprising:

a processor, memory accessible to the processor, and a data connection;

a machine-readable, executable routine containing instructions to cause the processor to send requests from the media player to the media source via the data connection for one or more serially identified media data elements representing an audio or video program, each media data element comprising a digitally encoded portion of the audio or video program and having a playback rate, each request specifying one or more serial identifiers of the media data elements requested;

a machine-readable, executable routine containing instructions to cause the media player to receive each of the requested media data elements via the data connection, wherein the data connection has a data rate more rapid than the playback rate of the media data elements, and each received media data element is received at a rate as fast as the data connection between the media source and the media player allows;

a machine-readable, executable routine containing instructions to cause the processor to store the received media data elements in the memory of the media player;

a machine-readable, executable routine containing instructions to cause the received media data elements to be played in series from the memory of the media player; and

a machine-readable, executable routine containing instructions to cause the processor to, as the received media data elements are played, send additional requests for subsequent media data elements for storage in the memory of the media player as required to maintain about a predetermined number of media data element in the memory of the media player during playing.

7. The device of claim **6**, further comprising a machine-readable, executable routine containing instructions to cause the processor to maintain in the memory of the media player a record identifying the last media data element received and stored by the media player.

8. The device of claim **6**, wherein the serial identifiers are sequential.

9. The device of claim **6**, wherein the receiving is via a reliable transmission protocol.

10. The device of claim **9**, wherein the reliable transmission protocol is TCP.

11. A computer program product for receiving and playing an audio or video program by a media player having a processor and a memory accessible to the processor, from a remote media source via a data connection over the Internet, the computer program product comprising a non-transitory computer readable storage medium having program instructions embodied therewith, the program instructions comprising:

instructions executable to cause the processor to send requests from the media player to the media source via the data connection for one or more serially identified media data elements representing the audio or video program, each media data element comprising a digitally encoded portion of the an audio or video program and having a playback rate, each request specifying one or more serial identifiers of the media data elements requested;

instructions executable to cause the processor to receive each of the requested media data elements via the data connection, wherein the data connection has a data rate more rapid than the playback rate of the media data

elements, and each received media data element is received at a rate as fast as the data connection between the media source and the media player allows;

instructions executable to cause the processor to store the received media data elements in the memory of the media player;

instructions executable to cause the received media data elements to be played in series from the memory of the media player; and

instructions executable to cause the processor to, as the received media data elements are played, send additional requests for subsequent media data elements for storage in the memory of the media player as required to maintain about a predetermined number of media data elements in the memory of the media player during playing.

12. The computer program product of claim **11**, further comprising instructions executable to cause the processor to maintain in the memory of the media player a record identifying the last media data element received and stored by the media player.

13. The computer program product of claim **11**, wherein the serial identifiers are sequential.

14. The computer program product of claim **11**, wherein the computer program product is provided as a software application for the media player.

15. The computer program product of claim **11**, wherein the computer program product is incorporated in the media player.

16. The computer program product of claim **11**, wherein the receiving is via a reliable transmission protocol.

17. The computer program product of claim **16**, wherein the reliable transmission protocol is TCP.

* * * * *