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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
11 October 2012 (11.10.2012)

WIPO | PCT

(10) International Publication Number
WO 2012/138660 A2

(51) **International Patent Classification**:
H04N 21/238 (2011.01)

(21) **International Application Number**:
PCT/US2012/032010

(22) **International Filing Date**:
3 April 2012 (03.04.2012)

(25) **Filing Language**: English

(26) **Publication Language**: English

(30) **Priority Data**:
61/473,085 7 April 2011 (07.04.2011) US

(71) **Applicant** *(for all designated States except US)*: **ACTIVEVIDEO NETWORKS, INC.** [US/US]; 333 W. San Carlos Street, Suite 400, San Jose, CA 95110 (US).

(72) **Inventors; and**

(75) **Inventors/Applicants** *(for US only)*: **BROCKMANN, Ronald** [DE/NL]; Mediacentrum 310, Sumatralaan 45, NL-1217 GP Hilversum (NL). **DEV, Anuj** [NL/NL]; Mediacentrum 310, Sumatralaan 45, NL-1217 GP Hilversum (NL). **HIDDINK, Gerrit** [NL/NL]; Mediacentrum 310, Sumatralaan 45, NL-1217 GP Hilversum (NL). **DAHLBY, Joshua** [US/US]; 1101/2 S. Navarra Drive, Scotts Valley, CA 95066 (US). **PAVLOVSKAIA, Lena, Y.** [US/US]; 10355 El Prado Way, #B, Cupertino, CA 95014 (US).

(74) **Agents**: **SUNSTEIN, Bruce D.** et al.; Sunstein Kann Murphy & Timbers LLP, 125 Summer Street, Boston, MA 02110 (US).

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

(84) **Designated States** *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) **Title**: REDUCTION OF LATENCY IN VIDEO DISTRIBUTION NETWORKS USING ADAPTIVE BIT RATES

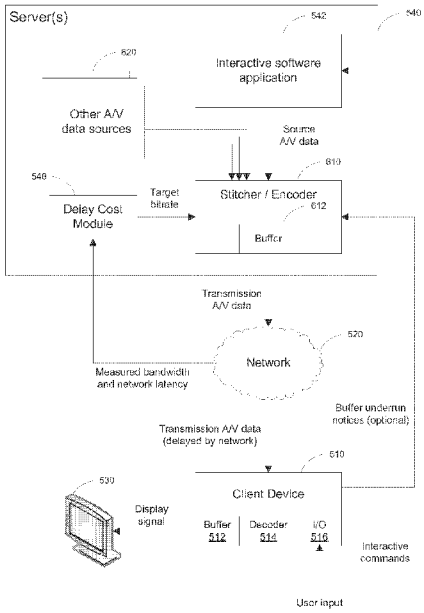

FIG. 6

(57) **Abstract**: Systems and methods are provided for reducing and controlling playback latency in an unmanaged, buffered data network, A delay cost function is determined, the function representing the effect of playback latency on end user experience. An encoder transmits audiovisual data through the network to a client device. Network latency is measured, and the delay cost function is evaluated to establish an encoding bitrate for the encoder. The encoding of the audiovisual data is altered in response to dynamic network conditions, thereby controlling end-to-end playback latency of the system, which is represented by the playout length of data buffered between the encoder and the client device.

WO 2012/138660 A2

Published:

— *without international search report and to be republished upon receipt of that report (Rule 48.2(g))*

Attorney Docket: 1436/A10WO

Reduction of Latency in Video Distribution Networks Using Adaptive Bit Rates

Cross-Reference to Related Application

[0001] This application claims the benefit of U.S. Provisional Application No. 61/473,085, filed April 7, 2011, the contents of which are herein incorporated by reference in their entirety.

Technical Field

[0002] The present invention relates to reducing playback latency in video distribution networks, and more particularly to adjusting an audiovisual encoding bitrate based on a detected network latency using a delay cost function that is indicative of the effect of playback latency on an end user experience.

Background Art

[0003] Interactive television services provide a television viewer the ability to interact with their television. Such services have been used, for example, to provide navigable menuing systems and ordering systems that are used to implement electronic program guides and on-demand and pay-per-view program reservations without the need to call a television provider. These services typically employ an application that is executed on a server located remotely from the viewer. Such servers may be, for example, located at a cable television headend. The output of the application is streamed to the viewer, typically in the form of an audiovisual MPEG Transport Stream. This allows the stream to be displayed on virtually any client device that has MPEG decoding capabilities, including a television set top box. The client device allows the user to interact with the remote application by capturing keystrokes and passing these back to the application.

[0004] The client and the server are, in cable deployments, separated by a managed digital cable-TV network that uses well-known protocols such as ATSC or DVB-C. Here, 'managed' means that any bandwidth resources required to provide these services may be

reserved prior to use. Once resources are allocated, the bandwidth is guaranteed to be available, and the viewer is assured of receiving a high-quality interactive application experience.

[0005] In recent years, audio-visual consumer electronics devices increasingly support a Local Area Network (LAN) connection, giving rise to a new class of client devices: so-called "Broadband Connected Devices", or BCDs. These devices may be used in systems other than the traditional cable television space, such as on the Internet. For example, consider Figure 1, in which a client device 110 (such as a Blu-ray player) implements a client application 112 to deliver audiovisual applications streamed over a public data network 120 from an audiovisual application streaming server 130 to a television 140. A user may employ a remote control 142 in conjunction with the client device 110 to transmit interactive commands back to the application streaming server 130, thereby controlling the content interactively.

[0006] However, because public data networks are not managed in the same way that private cable systems are, challenges arise. The transport protocols that are commonly used on the open Internet (such as TCP or RTSP) do not support bandwidth reservation. Since bandwidth cannot be guaranteed, the application server is not assured that the network connection can deliver the requested bandwidth. The actual throughput of an Internet connection can vary from second to second depending on many factors, including: network congestion anywhere between the application server and the client device; high-throughput downloads or uploads sharing the same physical internet connection as the client device (e.g. an ADSL line); mechanisms at lower (data link) layers that introduce delay, for example Adaptive Retransmission (ARQ) mechanisms in (wireless) access protocols; lost packets at any link between the client and the server; Transmission Control Protocol (TCP) state and more specifically TCP congestion window size; and reordering of packets caused by any link between the client and the server. To the server streaming the data to the client device, these factors all manifest themselves as fluctuations in actual achieved throughput. Small fluctuations can be addressed by using sufficient buffering, however buffering causes larger end-to-end delays (the time between the moment a user pressing a remote control button, and the moment that the screen update as a result of the key press has been rendered on the user's screen). Delays as short as five seconds may result in an unpleasant viewer experience in some applications such as an electronic program guide, while delays of even one-half second may be extremely noticeable in high-performance gaming applications. Further, the use of such buffering cannot compensate for large fluctuations in throughput.

[0007] Figs. 2-4 illustrate an example of the type of end-to-end playback latency in a typical network system, such as that of Fig. 1, during a transient network outage. There are three sources of playback latency: a server buffer that represents a source of pre-transmission latency; a network buffer that represents transmission latency in the public data network; and a client buffer that represents post-transmission latency in the client device before the audiovisual data are shown. Because of these sources of latency, at a time T1, as the application streaming server 130 generates data for display, the client device 110 is displaying data generated at an earlier time T0. The data that have been generated but not yet viewed are distributed in the three buffers awaiting display. The data themselves are visually represented and discussed in terms of video frames for ease of understanding.

[0008] Fig. 2 shows the system operating normally at time T1, just before a network outage occurs between the public data network 120 and the client device 110. Fig. 3 shows the system at a time T2 that is 200 ms later, at the end of the network outage. Fig. 4 shows the system at a time T3 that is another 30 ms later (that is, 230 ms after the start of the outage), after the network has had a chance to transmit some of its buffered data to the client device. These figures are now described in more detail.

[0009] More particularly, Fig. 2 shows a server buffer, a network buffer, and a client buffer at a time T1. This network is operating in equilibrium: on average, application server 130 generates one frame of video data in the length of time that each frame of video is displayed on the client device 110 (typically, 1/30 of a second). There are 180 ms of buffered playout data in this Figure: 50 ms in the server buffer, 80 ms in the network buffer, and 50 ms in the client buffer. To be even more specific, the 50 ms of data in the server buffer represent data generated in the 50 ms prior to time T1. Thus, the server buffer contains data spanning the playback range (T1 – 50 ms, T1), and the first frame of data in the server buffer was generated at time T1 – 50 ms, as indicated. The data in the network buffer were generated over the 80 ms prior, and therefore span the playback range (T1 – 130 ms, T1 – 50 ms). The data in the client buffer were generated over the 50 ms prior, and span the playback range (T1 – 180 ms, T1 – 130 ms). Therefore, the display device 140 is playing out the video frame for T0 = T1 – 180 ms from the top of the client buffer. Assuming that the system continues operating with these latencies, and assuming that the application server can generate a frame instantly in response to user input, a keystroke entered using remote control 142 at time T1 will cause a visible reaction on the display

device 140 at time T1 + 180 ms. That is, the keystroke will have a visible effect as soon as the buffered frames have emptied out of the three buffers onto the display, and the new frame can be displayed. Thus, the system as shown includes a response time of just under two tenths of a second. This delay is barely noticeable for an electronic program guide application.

[0010] Continuing the example, suppose a network outage between the network buffer and the client occurs immediately after the time T1, and lasts for 200 ms. At this point, the buffers may appear as in Fig. 3. Here, the client has drained its 50 ms of data, and playout is paused at T1 – 130 ms. It has been paused there for 150 ms (i.e., the amount of time that has elapsed for which it has not received any data). Meanwhile, the server has generated 200 ms of additional audiovisual data for playback. Based on the particular bandwidths available in the network during the outage, only 110 ms of playback have been sent to the network. Thus, the network buffer has 190 ms of stored data: 110 ms of new data, plus the 80 ms that it had at the beginning of the outage. No data have been sent from the network buffer to the client buffer, so the network buffer has data for 190 ms of playback in the range (T1 – 130 ms, T1 + 60 ms). In Fig. 2 the server buffer had data that began at T1 – 50 ms. In the intervening 200 ms, 110 ms of data have passed through the buffer and 90 ms of data have accumulated there. These 90 ms are in addition to the 50 ms already there, so the server buffer now has 140 ms of playback data. These data span the range (T1 + 60 ms, T1 + 200 ms).

[0011] The 200 ms of playback generated by the server during the outage have been buffered in the network. The 200 ms of data are split between the server buffer (90 ms of increase) and the network buffer (110 ms of increase). The total non-client buffering has increased from only 130 ms (about 1/8 of a second) to 330 ms (about 1/3 of a second).

[0012] Thus, after the outage has been resolved, an additional 200 ms of data will be buffered in the system. This can be seen in Fig. 4, which corresponds to the state of the system 30 ms after the outage. In these 30 ms, the network provided enough bandwidth to the client to transfer 50 ms of playback data, which are seen in the client buffer. These data span the range (T1 – 130 ms, T1 – 80 ms). The client has just received enough data to safely resume playback, so playback is resumed at T1 – 130 ms. Looking at the network buffer, 50 ms of playout data have been sent to the client, but 50 ms of playout data have been received from the server, so the network buffer still has 190 ms of data, now spanning the range (T1 – 80 ms, T1 + 110 ms). Meanwhile, the server has generated an additional 30 ms of data and transmitted 50 ms of data to

the network, so the server buffer has 120 ms of data spanning the range (T1 + 110 ms, T1 + 230 ms).

[0013] From these figures it is clear that a buffer underrun at the client can lead to playback latency buildup. The system of Fig. 2 had 130 ms of end-to-end delay, but by the end of Fig. 4 when playback resumed, an additional 230 ms of delay had been introduced into the system. Thus, in Fig. 4, there are 130 ms + 230 ms = 360 ms of total latency in the system, distributed between the three buffers. This playback latency buildup occurs for each client buffer underrun, and such buildups are cumulative. This is a highly undesirable situation for interactive applications.

[0014] The prior art does not adequately solve this problem. The client cannot simply skip individual frames because typical encoding schemes, such as MPEG, may encode each frame based on the data contained in previous and subsequent frames. The client could skip to its next intracoded frame, but these frames may be infrequent, and in any event such a strategy might be jarring for the viewer watching the stream. The server cannot pause frame generation, since it has no indication of the playout problems at the client. A new approach is therefore needed.

Summary of the Embodiments

[0015] Various embodiments of the invention optimize playback latency across an unmanaged network as a function of measured network latency and available network bandwidth. Users tolerate variations in playback latency differently for different applications, such as interactive channel menus and program guides, video games, billing systems and the like. Thus, in accordance with various embodiments, these variations are captured in a delay cost function, and playback latency is optimized based on the application. The delay cost function represents, in a way, the effect of playback latency on the user experience.

[0016] Ideally, zero latency across all applications would be optimal, but this is not possible in practice because the data network introduces network latencies that are unknown in advance, and uncontrollable by the application. Thus, the systems and methods disclosed herein take measurements of the data network, and adjust playback latency accordingly. In some embodiments, playback latency is adjusted by varying the amount of new frame data being

placed into the network for transmission. In others, playback latency is adjusted by notifying the application generating the source data, so that the source data themselves are modified.

[0017] Thus, in a first embodiment of the invention there is provided a method of controlling playback latency associated with transmission of source audiovisual data through an unmanaged, buffered data network. The method includes: encoding the source audiovisual data, according to an encoding bitrate, into transmission audiovisual data; transmitting the transmission audiovisual data to a client device through the data network; calculating a delay cost function based on a network latency associated with the data network; and altering the encoding of the source audiovisual data based on the calculated delay cost function.

[0018] Encoding may be performed according to an MPEG standard. The client device may be, among other things, a television set top box, a television, a personal computer, a tablet computer, a smartphone, or an optical disc player such as a Blu-ray or DVD player or game console. The unmanaged, buffered data network may be at least one of a cable data network, a broadcast wireless data network, a point-to-point wireless data network, a satellite network, and a portion of the Internet. Or, the unmanaged, buffered data network may be coupled to a managed data network that is capable of providing interactive television signals.

[0019] In a related embodiment, transmitting includes dividing the data of each video frame into at least one frame portion. In this embodiment, the method further calls for waiting to receive, from the client device, for each portion, an acknowledgement that the portion has been received by the client device, wherein the delay cost function is based on a length of time ("delay") between the completion of the encoding of a portion and receipt from the client device of the acknowledgement of the portion. The delay cost function may be calculated as

$$\mathbf{cost} = \alpha + \lambda^{*}(\mathbf{delay} - \mathbf{rtt\_min})^{\gamma},$$

where α is a number that represents a minimum cost to transmit data, λ is a number that indicates a scaling factor of the delay cost function, γ is a number that indicates a curvature of the delay cost function, and "rtt_min" is a minimum round trip time associated with the data network. In a different embodiment, the bitrate cost is not based on a delay, but based on variation of a round trip time ("rtt") that is associated with the data network. In this case, the delay cost function may be calculated as $\mathbf{cost} = \alpha + \lambda^{*}(\mathbf{rtt} - \mathbf{rtt\_min})^{\gamma}$.

[0020] In a further embodiment, transmitting is performed in accordance with the Transmission Control Protocol ("TCP"), and the method calls for calculating an estimated available bandwidth as

estimate = 8 * mss * cwnd / rtt,

where "mss" is a TCP Maximum Segment Size, "cwnd" is a TCP Congestion Window Size, and "rtt" is a round trip time associated with the data network. An encoding bitrate is established, and is equal to the value of the ratio of "estimate" to the delay cost function. Altering the encoding of the source audiovisual data may include altering the encoding bitrate to be no greater than the established encoding bitrate. If the source audiovisual data includes source audio data having an audio bitrate and source video frames, altering the encoding of the source audiovisual data may instead include altering a video encoding bitrate so that the sum of the audio bitrate and the video encoding bitrate is no greater than the established encoding bitrate.

[0021] In another embodiment, the source audiovisual data are generated by an interactive software application according to the established encoding bitrate, and the method further includes notifying the application that the encoding bitrate has been increased or decreased. The application responds by altering the source audiovisual data that it produces, including adjusting various screen objects and optimizing the generation of dynamic elements. When the established encoding bitrate is lower than a given threshold defined by the application, the application may alter the source audiovisual data by not generating a transparent screen object. If the source audiovisual data include a graphical user interface, then when the established encoding bitrate is lower than a given threshold defined by the application, the application may alter the source audiovisual data by postponing the generation of a dynamic screen region in the graphical user interface. In yet another embodiment, the method includes pausing the encoder when the delay cost function falls below a given threshold.

[0022] System embodiments and computer program product embodiments that perform the above methods are also disclosed.

Brief Description of the Drawings

[0023] The foregoing features will be more readily understood by reference to the following detailed description, taken with reference to the accompanying drawings, in which:

[0024] Fig. 1 is a system diagram showing an environment in which some embodiments of the invention may be used;

[0025] Fig. 2 is a latency diagram showing three sources of end-to-end latency in a typical network system that is performing normally;

[0026] Fig. 3 is a latency diagram showing latencies in the system of Fig. 2 a short time later, after a blockage has developed between the network buffer and the client device;

[0027] Fig. 4 is a latency diagram showing latencies in the system of Fig. 3 a short time later, after the blockage has been corrected;

[0028] Fig. 5 is a functional diagram showing relevant functional components of a particular embodiment of the invention;

[0029] Fig. 6 is a functional diagram showing relevant functional components of an alternate embodiment;

[0030] Fig. 7 is a flowchart showing the processes associated with a method of reducing latency in accordance with an embodiment of the invention;

[0031] Fig. 8 is a graph of a delay cost function used to respond to detected latency in one embodiment;

[0032] Fig. 9 is a graph of a delay cost function used to respond to detected latency in an alternate embodiment;

[0033] Fig. 10 is a flowchart showing four processes associated with a method of detecting a client buffer underrun in another embodiment; and

[0034] Fig. 11 shows the relationship between two timestamps used to detect the client buffer underrun.

Detailed Description of Specific Embodiments

[0035] As used in this description and the accompanying claims, the following terms shall have the meanings indicated, unless the context otherwise requires:

[0036] **Playback latency** refers to a delay, experienced by the user of an interactive audiovisual application displayed on a display device, between an input provided by the user and a reaction of the audiovisual information. In more concrete terms, one kind of playback latency is the time between pressing a "channel menu" key on a television remote control and the actual appearance of the menu on a television screen. Assuming that the application generating the

displayed video reacts instantly to the receipt of the key press, the television nevertheless does not react immediately, because some video data is in transit between the application and the television, buffered in various locations. This buffered data represents the playback latency. Playback latency is typically measured in frames of video or audio data.

[0037] **Network latency** refers to a delay resulting from transmission of data through a data network. Because data networks are made of physical components, and because these components can only transmit information at a maximum physical speed, there is a delay caused by the transit time of data through a network. Additionally, if the receiving system cannot receive the data as fast as the sending system is generating it, or for other reasons, the data network itself may buffer data, slowing its transit. These delays give rise to network latency. Network latency is typically measured in milliseconds (ms).

[0038] **Bandwidth** refers to the rate at which data may be transmitted through a data network. Bandwidth is independent of latency. For example, an interplanetary data network may have large network latency but also a high bandwidth, while a serial cable connecting a keyboard to a computer may have a very low latency but also a low bandwidth. Public data networks like the Internet generally strive to have a high bandwidth and a low network latency; that is, the ability to move 'as much data' as possible, 'as quickly' as possible.

[0039] A **managed** data network is one that can guarantee a particular bandwidth will be available. Typical managed networks, such as cable networks based on ATSC or QAM, use bandwidth reservation protocols, such as the Resource Reservation Protocol (RSVP), to guarantee availability. By contrast, an **unmanaged** data network is one that cannot make such a guarantee. Most public data networks are unmanaged.

[0040] Various embodiments of the invention optimize playback latency across an unmanaged network as a function of available network bandwidth and measured network latency. Users tolerate variations in playback latency differently for different applications, such as interactive channel menus and program guides, video games, billing systems and the like. Thus, in accordance with various embodiments, these variations are captured in a delay cost function, one for each application, and playback latency is optimized based on the application.

[0041] Ideally, zero latency across all applications would be optimal, but this is not possible in practice because the data network introduces network latencies that are unknown in advance, and uncontrollable by the application. Thus, the systems and methods disclosed herein

carefully balance the amount of playback latency generated by a streaming server, as a function of the current state of the network. In particular, these systems take measurements of the data network, and adjust playback latency accordingly. In some embodiments, playback latency is adjusted by varying the amount of new frame data being placed into the network for transmission. In others, playback latency is adjusted by notifying the application that is generating the source data, so that the source data themselves are modified.

[0042] In addition to addressing network latency, different embodiments account for variations in the network bandwidth. Audiovisual data are produced at one bitrate, transmitted through the data network at a different and uncontrolled bitrate (namely the instantaneously available network bandwidth), and consumed at a third bitrate. In addition to balancing the playback latency, systems and methods disclosed herein also carefully balance the amount of data being put into the data network against fluctuations in network bandwidth to avoid client data underruns. While client data underruns are preferably avoided, various methods for dealing with them are also discussed.

[0043] Fig. 5 is a functional diagram showing relevant functional components of a particular embodiment of the invention. Similar to Fig. 1, there is a client device 510, a data network 520, a display device 530, and one or more servers 540 that produce audiovisual data for display on the display device. The audiovisual data may be, for example MPEG encoded data.

[0044] The client device 510 may include, among other things, a television set top box, a broadband connected television, a personal computer, a tablet computer, a smartphone, or an optical disc player such as a DVD player or Blu-ray player (either as a standalone unit or as part of a video game console). The client device 510 should be able to receive the audiovisual data and convert it into video and audio signals that may be shown and/or heard on display device 530. Thus, client device may be a 'thin client', as that phrase is known in the art. A buffer 512 in the client device corresponds to the client buffer of Figs. 2-4, and is used to delay frames of data until the proper time to transmit them to the display device 530 for immediate display. A decoder 514 receives audiovisual data (which has been encoded according to an encoding, such as MPEG) and decodes it into video and audio data that are placed into the buffer 512. An input/output unit 516 receives user input commands (for example, from a remote control) and transmits them to an interactive software application, in the one or more servers 540, to control

the received video and audio. The buffer 512, decoder 514, and I/O unit 516 may be implemented in hardware, software, or a combination of these.

[0045] The data network 520 may include a cable data network, a broadcast wireless data network, a point-to-point wireless data network, a satellite network, or a portion of the Internet. In some embodiments, the data network 520 is coupled to a managed data network that is capable of providing interactive television signals to a viewer. Thus, for instance, the data network 520 may be the Internet, and a managed cable television network, including a cable headend, may be interposed between the data network 520 and the client device 510. In this embodiment, the servers 540 are remote from the cable headend, yet are controlled directly by the viewer. This kind of arrangement is described in more detail in U.S. Patent Application 10/253,109, filed September 24, 2002 and titled "Interactive Cable System with Remote Processors," the contents of which are incorporated herein by reference in their entirety. Alternatively, a managed cable television network may be interposed between the data network 520 and servers 540. This embodiment corresponds, for example, to a situation in which a cable company transmits interactive signals to neighborhood signal distribution nodes, but the nodes themselves aggregate bandwidth to individual homes or businesses in an unmanaged fashion. In this embodiment, data network 520 corresponds to the "last mile" of connectivity, as that phrase is known in the art. In such systems, bandwidth to an individual viewer cannot be guaranteed (managed) as a consequence of performing the aggregation, even if data network 520 is privately owned by the cable company. In a combined embodiment, the server(s) 540 are remote from a cable headend and the last mile is unmanaged, so the system has two sources of unmanaged network latency. Variations of the network topology, in accordance with other embodiments of the invention, may be contemplated by those having ordinary skill in the art. The data network 520 is thus unmanaged, in that bandwidth throughput cannot be guaranteed. The data network 520 is also a source of network latency, represented by the network buffer of Figs. 2-4.

[0046] The display device 530 is configured to convert audiovisual signals into images and sounds. The display device 530 may be, among other things, a television, a computer monitor, or a smartphone display. However, it is not intended to limit the scope of the invention by enumerating these various embodiments, and a person of ordinary skill in the art may see how to adapt other technologies to meet the requirements of the display device 530.

[0047] The box 540 denotes a system of one or more computers (servers) that generate transmission audiovisual data in accordance with an embodiment of the invention. The functional components of the element 540 include an interactive software application 542, an encoder 544, and a delay cost module 548. Their representation as a single element does not limit the scope of the invention, as these functional components may be implemented in a variety of hardware and/or software environments using only one computing processor or using multiple processors, or using ASICs or FPGAs.

[0048] The interactive software application 542 is a hardware or software component that produces audiovisual data, and alters that audiovisual data in response to receiving interactive commands from the client device 510, as previously described. The application 542 may provide, for example, an electronic program guide or other graphical user interface, a billing system, an authorization mechanism, a video game, a web browser, an email client, a music browsing and purchasing application, Internet access, or another type of interactive application. The source audiovisual data may be raw frames of uncompressed video and/or audio data, or they may be compressed according to a compression algorithm known in the art.

[0049] The application 542 sends the source audiovisual data, according to a source bitrate, to an encoder 544. The encoder 544 encodes the source audiovisual data, according to an encoding bitrate, to produce transmission audiovisual data. The transmission audiovisual data are formatted for distribution according to a particular encoding standard, such as MPEG, that the client device 510 (and more particularly, decoder 514) is capable of decoding for playback.

[0050] Transmission audiovisual data are stored in a transmission buffer 546 that corresponds to the server buffer of Figs. 2-4. This buffer 546 may be used to store data for transmission in the event that the network 520 has insufficient bandwidth to transmit data at the encoding bitrate used by the encoder 544. Data are added to the buffer 546 at the instantaneous encoding bitrate, and removed from the buffer 546 at the instantaneous network transmission bitrate. A ring buffer may be used for this purpose.

[0051] In the case that the encoding output bitrate is greater than the network throughput, buffer 546 stores the data that cannot be transmitted immediately. In general, it is very difficult to determine when sufficient network throughput is available. The underlying transport protocol, which is typically TCP, can best be seen as elastic. That is, when less data is offered to TCP for transmission, it will allow other competing connections to take a larger share. Similarly, when

more data is offered to TCP for transmission, it will take more bandwidth. However, TCP cannot take more throughput than it rightfully may claim according to the fair sharing principles that have been built into its algorithms. The only effect that is observable to a server 540 is that when more data is queued for transmission than TCP can allow (as a collective of all connections competing for the same bandwidth), the data will queue up in the transmit buffer.

[0052] As will be appreciated by those having skill in the art of video encoding, different frames of video have different sizes based on their content, and sometimes based on the content of other video frames. Thus, the instantaneous encoding bitrate of the encoder 544 may vary over time, and may differ from an established (target) encoding bitrate. Similarly, the instantaneous available bandwidth of network 520 may vary over time. However, as long as buffer 546 does not completely empty, the rate of data being transmitted to data network 520 over time should converge to the established encoding bitrate.

[0053] The amount of playback latency in the buffer 546 should be inversely related to the encoding bitrate, for two reasons. First, if the buffer 546 starts to fill up, further queuing should be avoided because it introduces additional playback latency that manifests as 'sluggishness' in the user experience. Second, if the buffer 546 often does not hold any data at all because the data network 520 has extra throughput, then the server(s) 540 should try to grab a bit more bandwidth from other, competing TCP connections in the data network 520 by requesting transmission of playback frames having more data, thereby increasing the visual quality of the video.

[0054] Thus, in accordance with various embodiments of the invention, a bitrate control algorithm monitors the amount of data that is queuing up, and establishes the encoding bitrate of encoder 544. The encoding bitrate for the encoder 544 changes as a function of a delay cost, and a delay cost module 548 is employed to calculate this function. The delay cost function may be broadly viewed as a measure of the cost of increased playback latency to an end user experience. This cost varies from one application to the next. Applications for which end users are sharply intolerant of playback latency beyond a small number of frames, such as video games, may have a delay cost function shaped as in Fig. 8 (which is described in more detail below). Applications for which end users are relatively tolerant of delay in response to key presses, such as an interactive program guide, may have a delay cost function shaped as in Fig. 9 (also described below). Returning to Fig. 5, the delay cost module 548 is used to establish an encoding bitrate

for the encoder 544 (and, in some embodiments, to interactive software application 542). Thus, as the measured network latency in the network 520 increases, the established encoding bitrate decreases, restricting the flow of new data (and new playback latency) into the system.

[0055] Encoding bitrates that are less than the playback bitrate cause new audiovisual data to be produced by the server 540 slower than they are consumed by the client device 510. If this behavior is prolonged, it will cause the buffer 512 to empty. If left uncorrected, this condition will manifest itself on the display device 530 as a frozen image, which may be viewed by an end user as even more unpleasant than excess latency.

[0056] Fig. 6 is a functional diagram showing relevant functional components of an alternate embodiment. This embodiment is similar to that shown in Fig. 5, except that the encoder has been replaced by a stitcher / encoder 610. An example of a stitcher / encoder as known in the art may be found in U.S. Patent Application 12/008,697, filed January 11, 2008, the contents of which are incorporated herein by reference in their entirety. A stitcher is a hardware or software functional module that acts as a multiplexer of sorts: it stitches several video or audio frames together to form a single output frame. Such stitchers are useful, for example, in an interactive program guide application. In this connection, interactive software application 542 produces a textual or graphical channel listing in response to interactive commands. The stitcher portion of stitcher / encoder 610 combines this listing with the video and/or audio of a channel preview whose data come from other audiovisual data sources 620. The encoder portion of stitcher / encoder 610 then encodes the stitched content according to the target average bitrate, and places it in the buffer 612 for transmission to the data network 520.

[0057] Fig. 7 is a flowchart showing the processes associated with a method of reducing and controlling playback latency in accordance with an embodiment of the invention. Such a flowchart may be, for example, embodied as computer program code that is executed in one or more of the functional modules of Figs. 5 and 6. In process 710, the method starts with encoding source audiovisual data into transmission audiovisual data. In process 720, the method requires transmitting the transmission audiovisual data to a client device through a data network. Process 730 includes calculating a delay cost function based on a network latency associated with the data network. In process 740, the method requires altering the stitching or encoding of the source audiovisual data based on the calculated delay cost function. As indicated in Fig. 7, these

processes may repeat, thereby providing a dynamic method of controlling playback latency in the unmanaged data network.

[0058] Various processes 730 for calculating the delay cost function are now discussed, with reference to Figs. 8 and 9. Fig. 8 is a graph of a delay cost function used to respond to detected network latency in one embodiment in which a user cannot tolerate much delay. Fig. 9 is a graph of a delay cost function used to respond to detected network latency in an alternate embodiment in which a user is more tolerant of delay.

[0059] The delay cost is a function of actual network latency, which may be measured using a number of different techniques. As is known in the art, TCP sends data in packets of a limited size. Therefore, when an application attempts to send more data than can be sent at once, the TCP sender breaks the data into sequenced packets, and sends the packets in sequence. The TCP receiver then acknowledges packets based on the sequence. MPEG video frames are often larger than the TCP packet size, so frames are often broken into multiple packets for transmission.

[0060] In one embodiment, the network latency is calculated in terms of MPEG video frame portions. Server(s) 540 divide the data of each video frame into at least one frame portion, but often into many portions. Each frame portion is transmitted to the data network 520 as a TCP packet, and eventually reaches the client device 510 which acknowledges it. The server(s) 540 wait to receive, from the client device, the acknowledgement for each frame portion sent through the network.

[0061] In this embodiment, the delay cost function is based on a length of time between the completion of the encoding of a frame portion and receipt of the acknowledgement of the frame portion (as a packet). This length of time is called the "**delay**". To calculate the delay, a list of all TCP packets having MPEG frame data is maintained, and packets that have not been acknowledged by the client TCP stack are counted toward the network latency measure. Frames for which no packets have been acknowledged add their full playout length (e.g. 1/30 of a second) to the network latency. A frame for which some packets have been acknowledged but not others is counted as a fraction of its full playout length, based on the number of bytes required to transmit that given frame and the number of bytes in the unacknowledged packets. As some frames require more data to encode than others, this calculation will differ from one frame to the next.

[0062] Once the delay has been calculated, the delay cost may be calculated. In one embodiment, the delay cost has the formula $\mathbf{cost} = \alpha + \lambda^*(\mathbf{delay} - \mathbf{rtt})^{\gamma}$. Here, **rtt** is the round trip time measured for the data network 520, discussed above. In some embodiments, the overall minimum round trip time **rtt_min** is used instead of the instantaneously measured round trip time **rtt**. The term (**delay** – **rtt**) represents the buffering delay at the server side, while the term (**delay** – **rtt_min**) represents this delay plus an amount of buffering (**rtt** – **rtt_min**) that is attributable to network delay. In some embodiments, this term (in either form) may be divided by a characteristic time, such as **rtt** or **rtt_min**, so that it becomes a dimensionless number prior to exponentiation. The minimum round-trip time is accepted as a given – it consists of network buffers and propagation delay that the server cannot control.

[0063] The shape of the cost function ensures that as the sender buffering delay increases, the server's generated bitrate is decreased. The parameter α specifies an offset with respect to the TCP estimated bitrate, and represents a minimum cost to transmit data. Selecting a value for α that is less than 1.0 will cause the server bitrate to be higher than the estimated available bitrate, which is useful to get TCP to 'stretch up' its maximum bandwidth. The positive parameter λ specifies a scaling factor, to allow the buffering delay to have a larger effect on the resulting bitrate. The parameter γ determines the shape of the cost function, and in particular its curvature. A γ value of 1.0 will provide a linear cost, so the target output bandwidth response will be purely hyperbolic. A γ value higher than 1.0 creates a steeply falling target bandwidth curve (and a more rapid response to detected network latency). Conversely, a γ value lower than 1.0 provides a more gradual response to detected network latency.

[0064] For example, Fig. 8 is a graph of a delay cost function used by the system to respond to detected network latency in one embodiment in which a user cannot tolerate much playback latency. For example, this curve may represent user tolerance in a video game application. This particular curve corresponds to an instantaneous estimated TCP bitrate of 3.75 Mbps, delay measured in tenths of a second, with $\alpha = 1$, $\lambda = 0.1$, and $\gamma = 4$. Thus, when the network is performing optimally with zero delay, the target output bitrate is 3.75 Mbps. When there is 0.1s of delay detected, the target output bitrate is 3.50 Mbps, because this level of latency is still acceptable. But when delay rises to 0.2s, the target output bitrate has fallen drastically to about 1.50 Mbps, and by 0.4s of delay, the transmission of new data has virtually stopped. Thus, fewer data accumulate in the server buffer than before the bitrate reduction, but they will be

timelier. This reduction in bitrate results in continued high responsiveness to user inputs, even if the quality of the resulting video is degraded. In an application that is intolerant of high playback latency, this is the proper tradeoff.

[0065] By contrast, Fig. 9 is a graph of a delay cost function used to respond to detected network latency in an alternate embodiment in which a user is more tolerant of playback latency, such as an interactive program guide. The instantaneous estimated TCP bitrate is still 3.75 Mbps, but now $\alpha = 1$, $\lambda = 0.3$, and most importantly, $\gamma = 1$. Since $\gamma = 1$, the curve has a hyperbolic shape, and the response to delay is more gradual. When 0.1s of delay is detected, the target output bitrate has fallen off to 2.90 Mbps. At 0.2s of delay, however, the target output bitrate is still at 2.40 Mbps, and at 0.4s of delay the bitrate is at 1.75 Mbps. These latter two numbers are much higher than in the previous example, because an end user of this application is much more tolerant of playback latency.

[0066] The algorithm discussed above does not take into account TCP's round-trip time (**rtt**) variations. On some occasions, especially when the data network has large network buffers, the observed round-trip time can become orders of magnitude larger than the minimum round-trip time **rtt_min** that is associated with the network. This is undesired since the **rtt** adds to the end-to-end network latency. Therefore, an alternate embodiment uses a cost function based on the instantaneous round-trip time, rather than the delay value calculated from MPEG frames. Thus, the delay cost function for this embodiment has the formula $\mathbf{cost} = \alpha + \lambda * (\mathbf{rtt} - \mathbf{rtt\_min})^{\gamma}$, where α, λ, and γ are selected accordingly.

[0067] Various processes 740 of altering the encoding of the source audiovisual data are now discussed. In general, there are two different ways to alter the encoding of the source audiovisual data: to change the encoding bitrate, and to change the source audiovisual data themselves. These approaches are explained in turn.

[0068] Generally speaking, the established encoding bitrate for the encoder is expressed as a ratio between an estimated available bandwidth and the delay cost function that corresponds to the currently measured latency. If TCP is used, the actual available bandwidth is unknown (i.e., TCP is "unmanaged"), although other protocols may be used that give an exact measure of available bandwidth. One formula for providing a TCP estimated bandwidth is **estimate = 8 * mss * cwnd / rtt**, where "**mss**" is the TCP maximum segment size, "**cwnd**" is the TCP congestion window size, and "**rtt**" is a round trip time associated with the data network. The

value of **rtt** may be measured using tools known in the art, such as the packet acknowledgement (ACK) mechanism. This formula operates on the principle that as network round trip time increases, TCP will buffer more data in its congestion window, and vice versa. As TCP stores data in bytes, multiplication by eight is necessary to yield a bitrate.

[0069] The actual reduction in encoder bitrate may be achieved in one of two ways. If the source audiovisual content is being encoded for the first time, the established encoding bitrate is passed directly to the encoder, which will encode the source audiovisual content accordingly. However, if the source data were pre-encoded, then the encoder will typically have access to a selection of different bitrates for the particular audiovisual content. When establishing a lower encoding bitrate, the playback latency in the server buffer is quantized, and each time a new quantized value is reached the next lower bitrate stream is selected. When establishing a higher bitrate, if a certain percentage of consecutive frames have been displayed on time, then the next higher bitrate stream is selected. The percentage is configurable to make the algorithm more 'adventurous'. For example, a particular video stored for on-demand playback may be pre-encoded at bitrates of 0.5 Mbps, 1.0 Mbps, and 2.0 Mbps, shown in Fig. 6 as data sources 620. Then, if the encoding bitrate is established at 1.5 Mbps, the stitcher / encoder 610 may stitch the 1.0 Mbps source data into the transmission audiovisual data stream, rather than the 2.0 Mbps source data. If the established encoding bitrate falls below 0.5 Mbps, or another given threshold, the encoder may be paused, so that no more data accumulate in the output buffer at all. By contrast, when the established encoding bitrate increases again above 2.0 Mbps, and (say) at least 75% of the frames were transmitted and displayed on time, the stitcher / encoder 610 may revert to using the higher-bitrate source data.

[0070] More subtle bitrate manipulations may be performed in other embodiments. For example, the source audiovisual data may include both audio and visual data, each having their own, separate bitrate. When the established encoding bitrate is reduced or increased, an encoder may reduce or increase the encoding bitrate of only the video portion of the source audiovisual data, while leaving the audio bitrate intact. If the video encoding bitrate is reduced enough, then the sum of the audio bitrate and the (reduced) video encoding bitrate may be no greater than the established encoding bitrate. In this way, a lower (or higher) target may be met without switching between two streams that have vastly different bitrates (and markedly different video qualities), as was discussed above. Further, by maintaining a constant audio bitrate, one may

establish a more precise encoding bitrate by determining a more accurate estimate of the available bandwidth in the data network 520. Systems and methods for doing so are taught in U.S. Application 12/651,203, filed December 31, 2009, the contents of which are incorporated herein by reference in their entirety.

[0071] Turning to a second way of altering the encoding of the source audiovisual data, in some embodiments of the invention, the source audiovisual data themselves are adapted to changing bitrates. This method of content adaptation corresponds to the dashed line in Fig. 5, whereby the delay cost module 548 notifies the interactive software application 542 that a new encoding bitrate has been established. In such embodiments, the interactive software application 542 responds not just to interactive commands from an end user, but also to changes in the encoding bitrate. In these embodiments, the application 542 creates source audiovisual data that may be encoded using a lower (or higher) bitrate, thereby cooperating with the encoder 544 that is trying to determine the best way to respond to the same, lower (or higher) established bitrate.

[0072] When an application is requested to generate a lower bitrate stream while maintaining high quality video, it can choose alternate video properties that require fewer bits to encode. This choice can still lead to satisfactory and attractive results, depending on the codec that is used for a given session. If MPEG is used, new screen elements are generally more expensive in terms of bitrate than moving elements, due to efficient encoding of motion vectors. Other seemingly simple effects, such as a fade in from black, are expensive (in MPEG2) because there is no compression primitive for recoloring or weighted prediction. However, MPEG4 has more facilities that allow such richer effects at low cost.

[0073] To reduce the encoding bitrate of the source audiovisual data to match the established target, the interactive software application 542 may use one or more of the following strategies. If it is generating a graphical user interface (GUI), it may select a less rich UI element. For example, instead of a cross-fade, fade-in from black or other color, the application 542 can generate the final element at once. Or the application can slide an object into view from the screen edge, rather than draw it all at once. Similarly, instead of an image having an alpha channel that smoothly blends into the background around the edges, an application can use a differently authored layout having an image without alpha channels (with smooth edges but no blending to the background). In this latter embodiment, a transparent or translucent screen object is not generated, in favor of an opaque object that requires fewer bits to encode.

[0074] An alternate strategy consists of postponing the production of selected screen regions until a later frame, if there are many updates in one frame and much fewer in later frames. For example, a GUI that includes a collection of buttons and a dynamic element (such as a channel preview) might update the dynamic element every other frame. While this reduces the effective frame rate of the dynamic element, it also reduces the actual size of the transmission audiovisual data. Another strategy includes showing only the end result (last frame) of an animation. Yet another strategy is to design a GUI background image to be a solid color rather than a complex pattern, to avoid having to re-render and retransmit the complex pattern when a pop-up window is removed from the screen. As a final strategy, for extremely low target average bitrates, the application could provide the source audiovisual data as a slideshow of images (i.e., JPEG or PNG images) rather than an MPEG stream.

[0075] There is another process 740 that may be employed, used when an underrun in the client buffer 512 underrun occurs. The use of the delay cost function described above is meant to avoid an actual client buffer underrun; thus, when one actually occurs, other measures are required. The alternative process 740 generally includes a cycle of four sub-processes that are shown in Fig. 10. The first process 1010 of the cycle involves estimating the size of the client buffer, and is described in more detail below. When the client buffer is estimated to be empty, the second process 1020 transitions from a 'streaming' state to a 'draining' state. This process includes pausing the encoder, as described above, to allow the network buffer to drain its playback latency. The third process 1030 estimates the size of the network buffer, and is described in more detail below. When the network buffer is estimated to be empty, the system is again ready to accept more playback latency, and the fourth process 1040 switches from the 'draining' state to the 'streaming' state. This process 1040 includes unpausing the encoder. The system shortly reaches equilibrium, and the cycle begins again.

[0076] There are two ways that the first process 1010 can detect a client buffer underrun, namely with or without direct client notice. Returning to Figs. 5 and 6, in some embodiments, the client device 510 includes logic to detect the existence or approach of a buffer underrun condition. This condition manifests itself, in a different context, in the familiar "buffering" message shown in some video players. When the condition is detected, the client device 510 signals the encoder 544 (or stitcher / encoder 610) to indicate that corrective measures are urgently required. This notice is shown by a dashed line in Figs. 5 and 6. When the client device

510 later signals that the buffer underrun has been cured, the encoder 542 unpauses to permit new data to reach the client.

[0077] In other embodiments, however, the client device 510 is unable to signal these underrun events, due to the modularity of software and shielding of software internals from developers. For example, often a set top box or a Blu-ray player incorporates a video decoding ASIC or system-on-chip, along with a software development kit (SDK) for the decoder. The SDK contains tools and software libraries to build a working product. Often the manufacturer of the client device can only access the ASIC via an application programming interface (API) that is provided with the SDK. If the API does not allow buffer underrun events to be signaled to other software modules, then this signal also cannot be sent from the client device 510 to the server.

[0078] In embodiments in which a client device 510 cannot signal an underrun in buffer 512 directly, the server estimates a buffer underrun event at the client. In one embodiment, the server(s) 540 detect when the server buffer is has more data than a given threshold. Based on the particular delay cost function used, the given threshold indicates that a network outage has occurred. If dynamic video is streamed, the threshold is set to a value that permits some server-side latency, because responsiveness to user input is not the overriding concern. However, if a user interface is streamed, the threshold is preferably set to zero. Thus, an interactive application will mostly remain in the 'draining' state, with no server-side playback latency. Furthermore, when draining occurs while streaming a user interface, the application 542 is not paused. This ensures minimum playback latency while the user is actively operating the application. Since no moving pictures are displayed, any 'hiccup' or stutter that would otherwise be noticed is not relevant here.

[0079] In one MPEG embodiment that does not rely on a buffer measure, the system uses the Program Clock Reference (PCR) with the Decoding Time Stamp (DTS) or Presentation Time Stamp (PTS). As is known in the art, the PCR is a datum in an MPEG-2 transport stream that provides a timestamp associated with the encoder's model of the presentation. The timestamp is used by the decoder 514 as a reference against which the other two timestamps are judged. The DTS provides a marker relative to the PCR that indicates a correct decoding time of a given audio or video frame, while the PTS indicates the correct playback time.

[0080] The relation between PCR and PTS in an ideal situation is shown in Fig. 11. Here, the encoder is encoding data using a playback latency of PTS – PCR. This value is equal to the end-to-end network latency, so the PTS time in the client device occurs just as the frame that should be displayed at that time arrives at the top of the client buffer. In prior art managed networks, the difference PTS – PCR is usually fixed, and is chosen based on a priori knowledge of (controllable) network latencies in the network path. Client devices often use PCR as a reference to determine playout timing in cable networks with fixed delay, but not when streaming over the open Internet. The use of the PTS in a public data network in this embodiment is advantageous, because it permits the detection of client buffer underruns in thin clients without explicit signaling, thus solving a different problem than its customary use for controlling display timing.

[0081] In this embodiment, the server detects a client underrun if the DTS or PTS for a frame has passed, but acknowledgement was not received for the last TCP packet of that frame. At the moment the DTS occurs, the client decoder 514 will be attempting to decode a particular frame to place in the buffer 512. Similarly, when the PTS occurs, the client device 510 will be attempting to display that frame. However, if the frame has not yet even been received by the client (as indicated by lack of packet acknowledgment), then the client buffer 512 must be empty or critically low. This condition may be detected by the server(s) 540, which may take corrective action even without receiving a signal from the client device 510 that a buffer underrun has occurred.

[0082] Corrective measures taken by a system that employs this method need not be limited to just pausing the encoder. When an underrun has occurred, the server may set a larger PTS – PCR difference, to allow the system more time to transfer the frames through the server, network, and client buffers. However, doing so would increase end-to-end playback latency. To reduce this playback latency, the server may receive buffer occupancy reports from the client. If the client buffer level is consistently above a certain threshold, then the server reduces the PTS – PCR difference accordingly.

[0083] The embodiments of the invention described above are intended to be merely exemplary; numerous variations and modifications will be apparent to those skilled in the art. All such variations and modifications are intended to be within the scope of the present invention as defined in any appended claims.

[0084] It should be noted that the logic flow diagrams are used herein to demonstrate various aspects of the invention, and should not be construed to limit the present invention to any particular logic flow or logic implementation. The described logic may be partitioned into different logic blocks (e.g., programs, modules, functions, or subroutines) without changing the overall results or otherwise departing from the true scope of the invention. Often times, logic elements may be added, modified, omitted, performed in a different order, or implemented using different logic constructs (e.g., logic gates, looping primitives, conditional logic, and other logic constructs) without changing the overall results or otherwise departing from the true scope of the invention.

[0085] The present invention may be embodied in many different forms, including, but in no way limited to, computer program logic for use with a processor (e.g., a microprocessor, microcontroller, digital signal processor, or general purpose computer), programmable logic for use with a programmable logic device (e.g., a Field Programmable Gate Array (FPGA) or other PLD), discrete components, integrated circuitry (e.g., an Application Specific Integrated Circuit (ASIC)), or any other means including any combination thereof. Hardware logic (including programmable logic for use with a programmable logic device) implementing all or part of the functionality previously described herein may be designed using traditional manual methods, or may be designed, captured, simulated, or documented electronically using various tools, such as Computer Aided Design (CAD), a hardware description language (e.g., VHDL or AHDL), or a PLD programming language (e.g., PALASM, ABEL, or CUPL).

[0086] The aforementioned computer program logic and programmable logic may be embodied in various forms, including, but in no way limited to, a source code form, a computer executable form, and various intermediate forms (e.g., forms generated by an assembler, compiler, linker, or locator). Source code may include a series of computer program instructions implemented in any of various programming languages (e.g., an object code, an assembly language, or a high-level language such as Fortran, C, C++, JAVA, or HTML) for use with various operating systems or operating environments. The source code may define and use various data structures and communication messages. The source code may be in a computer executable form (e.g., via an interpreter), or the source code may be converted (e.g., via a translator, assembler, or compiler) into a computer executable form.

[0087] The computer program may be fixed in any form (e.g., source code form, computer executable form, or an intermediate form) either permanently or transitorily in a tangible storage medium, such as a semiconductor memory device (e.g., a RAM, ROM, PROM, EEPROM, or Flash-Programmable RAM), a magnetic memory device (e.g., a diskette or fixed disk), an optical memory device (e.g., a CD-ROM), a PC card (e.g., PCMCIA card), or other memory device. The computer program may be fixed in any form in a signal that is transmittable to a computer using any of various communication technologies, including, but in no way limited to, analog technologies, digital technologies, optical technologies, wireless technologies (e.g., Bluetooth), networking technologies, and internetworking technologies. The computer program may be distributed in any form as a removable storage medium with accompanying printed or electronic documentation (e.g., shrink wrapped software), preloaded with a computer system (e.g., on system ROM or fixed disk), or distributed from a server or electronic bulletin board over the communication system (e.g., the Internet or World Wide Web).

What is claimed is:

1. A method of controlling playback latency associated with transmission of source audiovisual data through an unmanaged, buffered data network, the method comprising:

encoding the source audiovisual data into transmission audiovisual data;

transmitting the transmission audiovisual data to a client device through the data network;

calculating a delay cost function based on a network latency associated with the data network; and

altering the encoding of the source audiovisual data based on the calculated delay cost function.

2. The method of claim 1, wherein encoding is performed according to an MPEG standard.

3. The method of claim 1, wherein the client device comprises at least one of a television set top box, a television, a personal computer, a tablet computer, a smartphone, and an optical disc player.

4. The method of claim 1, wherein the unmanaged, buffered data network includes at least one of a cable data network, a broadcast wireless data network, a point-to-point wireless data network, a satellite network, and a portion of the Internet.

5. The method of claim 1, wherein the unmanaged, buffered data network is coupled to a managed data network that is capable of providing interactive television signals.

6. The method of claim 1, wherein transmitting includes dividing the data of each video frame into at least one frame portion, the method further comprising:

waiting to receive, from the client device, for each portion, an acknowledgement that the portion has been received by the client device, wherein the delay cost function is based on a length of time (hereinafter "**delay**") between the completion of the encoding of a portion and receipt from the client device of the acknowledgement of the portion.

7. The method of claim 6, wherein the delay cost function is

$$\mathbf{cost} = \alpha + \lambda^{*}(\mathbf{delay} - \mathbf{rtt\_min})^{\gamma},$$

where α is a number that represents a minimum cost to transmit data , λ is a number that indicates a scaling factor of the delay cost function, γ is a number that indicates a curvature of the delay cost function, and "**rtt_min**" is a minimum round trip time associated with the data network.

8. The method of claim 1, wherein the delay cost function is further based on variation of a round trip time (hereinafter "**rtt**") that is associated with the data network.

9. The method of claim 8, wherein the delay cost function is

$$\mathbf{cost} = \alpha + \lambda^{*}(\mathbf{rtt} - \mathbf{rtt\_min})^{\gamma},$$

where α is a number that represents a minimum cost to transmit data , λ is a number that indicates a scaling factor of the delay cost function, γ is a number that indicates a curvature of the delay cost function, and "**rtt_min**" is a minimum round trip time associated with the data network.

10. The method of claim 1, wherein transmitting is performed in accordance with the Transmission Control Protocol (hereinafter "TCP"), further comprising calculating an estimated available bandwidth as

$$\mathbf{estimate} = 8 * \mathbf{mss} * \mathbf{cwnd} / \mathbf{rtt},$$

where "**mss**" is a TCP Maximum Segment Size, "**cwnd**" is a TCP Congestion Window Size, and "**rtt**" is a round trip time associated with the data network, thereby establishing an encoding bitrate equal to the value of the ratio of "**estimate**" to the delay cost function, and wherein altering the encoding of the source audiovisual data includes encoding the source audiovisual data using the established encoding bitrate.

11. The method of claim 1, wherein transmitting is performed in accordance with the Transmission Control Protocol (hereinafter "TCP"), further comprising calculating an estimated available bandwidth as

$$\mathbf{estimate} = 8 * \mathbf{mss} * \mathbf{cwnd} / \mathbf{rtt},$$

where "**mss**" is a TCP Maximum Segment Size, "**cwnd**" is a TCP Congestion Window Size, and "**rtt**" is a round trip time associated with the data network, thereby establishing an encoding bitrate equal to the value of the ratio of "**estimate**" to the delay cost function, and wherein the source audiovisual data includes source audio data having an audio bitrate and source video frames, altering the encoding of the source audiovisual data including altering a video encoding bitrate so that the sum of the audio bitrate and the video encoding bitrate is no greater than the established encoding bitrate.

12. The method of claim 1, wherein the source audiovisual data are generated by an interactive software application according to the encoding bitrate, the method further comprising notifying the application that the encoding bitrate has been increased or decreased.

13. The method of claim 12, wherein, when the encoding bitrate is lower than a given threshold defined by the application, the application alters the source audiovisual data by not generating a transparent screen object.

14. The method of claim 12, wherein the source audiovisual data include a graphical user interface, and wherein when the encoding bitrate is lower than a given threshold defined by the application, the application alters the source audiovisual data by postponing the generation of a dynamic screen region in the graphical user interface.

15. The method of claim 1, further comprising pausing the encoder when the value of the delay cost function falls below a given threshold.

16. A tangible, computer-usable data storage medium on which is stored computer program code for instructing a computer system, that comprises at least one computing processor, to execute a method of controlling playback latency associated with transmission of source audiovisual data through an unmanaged, buffered data network, the program code comprising:

program code for encoding the source audiovisual data into transmission audiovisual data;

program code for transmitting the transmission audiovisual data to a client device through the data network;

program code for calculating a delay cost function based on a network latency associated with the data network; and

program code for altering the encoding of the source audiovisual data based on the calculated delay cost function.

17. The medium of claim 16, wherein encoding is performed according to an MPEG standard.

18. The medium of claim 16, wherein the client device comprises at least one of a television set top box, a television, a personal computer, a tablet computer, a smartphone, and an optical disc player.

19. The medium of claim 16, wherein the unmanaged, buffered data network includes at least one of a cable data network, a broadcast wireless data network, a point-to-point wireless data network, a satellite network, and a portion of the Internet.

20. The medium of claim 16, wherein the unmanaged, buffered data network is coupled to a managed data network that is capable of providing interactive television signals.

21. The medium of claim 16, wherein the program code for transmitting includes program code for dividing the data of each video frame into at least one frame portion, the medium further comprising:

program code for waiting to receive, from the client device, for each portion, an acknowledgement that the portion has been received by the client device, wherein the delay cost function is based on a length of time (hereinafter "**delay**") between the completion of the encoding of a portion and receipt from the client device of the acknowledgement of the portion.

22. The medium of claim 21, wherein the delay cost function is

$$\mathbf{cost} = \alpha + \lambda * (\mathbf{delay} - \mathbf{rtt\_min})^{\gamma},$$

where α is a number that represents a minimum cost to transmit data , λ is a number that indicates a scaling factor of the delay cost function, γ is a number that indicates a curvature of the delay cost function, and "**rtt_min**" is a minimum round trip time associated with the data network.

23. The medium of claim 16, wherein the delay cost function is further based on variation of a round trip time (hereinafter "**rtt**") that is associated with the data network.

24. The medium of claim 23, wherein the delay cost function is

$$\mathbf{cost} = \alpha + \lambda * (\mathbf{rtt} - \mathbf{rtt\_min})^{\gamma},$$

where α is a number that represents a minimum cost to transmit data , λ is a number that indicates a scaling factor of the delay cost function, γ is a number that indicates a curvature of the delay cost function, and "**rtt_min**" is a minimum round trip time associated with the data network.

25. The medium of claim 16, wherein transmitting is performed in accordance with the Transmission Control Protocol (hereinafter "TCP"), further comprising program code for calculating an estimated available bandwidth as

$$\mathbf{estimate} = 8 * \mathbf{mss} * \mathbf{cwnd} / \mathbf{rtt},$$

where "**mss**" is a TCP Maximum Segment Size, "**cwnd**" is a TCP Congestion Window Size, and "**rtt**" is a round trip time associated with the data network, thereby establishing an encoding bitrate equal to the value of the ratio of "**estimate**" to the delay cost function, and wherein altering the encoding of the source audiovisual data includes encoding the source audiovisual data using the established encoding bitrate.

26. The medium of claim 16, wherein transmitting is performed in accordance with the

Transmission Control Protocol (hereinafter "TCP"), further comprising program code for calculating an estimated available bandwidth as

estimate = 8 * mss * cwnd / rtt,

where "**mss**" is a TCP Maximum Segment Size, "**cwnd**" is a TCP Congestion Window Size, and "**rtt**" is a round trip time associated with the data network, thereby establishing an encoding bitrate equal to the value of the ratio of "**estimate**" to the delay cost function, and wherein the source audiovisual data includes source audio data having an audio bitrate and source video frames, altering the encoding of the source audiovisual data including altering a video encoding bitrate so that the sum of the audio bitrate and the video encoding bitrate is no greater than the established encoding bitrate.

27. The medium of claim 16, wherein the source audiovisual data are generated by an interactive software application according to the encoding bitrate, the medium further comprising program code for notifying the application that the encoding bitrate has been increased or decreased.

28. The medium of claim 27, wherein, when the encoding bitrate is lower than a given threshold defined by the application, the application alters the source audiovisual data by not generating a transparent screen object.

29. The medium of claim 27, wherein the source audiovisual data include a graphical user interface, and wherein when the encoding bitrate is lower than a given threshold defined by the application, the application alters the source audiovisual data by postponing the generation of a dynamic screen region in the graphical user interface.

30. The medium of claim 16, further comprising program code for pausing the encoder when the value of the delay cost function falls below a given threshold.

31. A system for controlling playback latency associated with transmission of source audiovisual data through an unmanaged, buffered data network, the system comprising:

an interactive application, executing on one or more computing devices, the interactive application being controlled by commands received from a client device to generate source audiovisual data;

an encoder for encoding the source audiovisual data into transmission audiovisual data according to an encoding bitrate;

a transmitter for transmitting the transmission audio data through the data network; and

a delay cost module for calculating a delay cost function based on a network latency associated with the data network, the delay cost module calculating the encoding bitrate based on the delay cost function and providing the encoding bitrate to the encoder.

32. The system of claim 31, wherein the client device comprises at least one of a television set top box, a television, a personal computer, a tablet computer, a smartphone, and an optical disc player.

33. The system of claim 31, wherein the unmanaged, buffered data network includes at least one of a cable data network, a broadcast wireless data network, a point-to-point wireless data network, a satellite network, and a portion of the Internet.

34. The system of claim 31, wherein the unmanaged, buffered data network is coupled to a managed data network that is capable of providing interactive television signals.

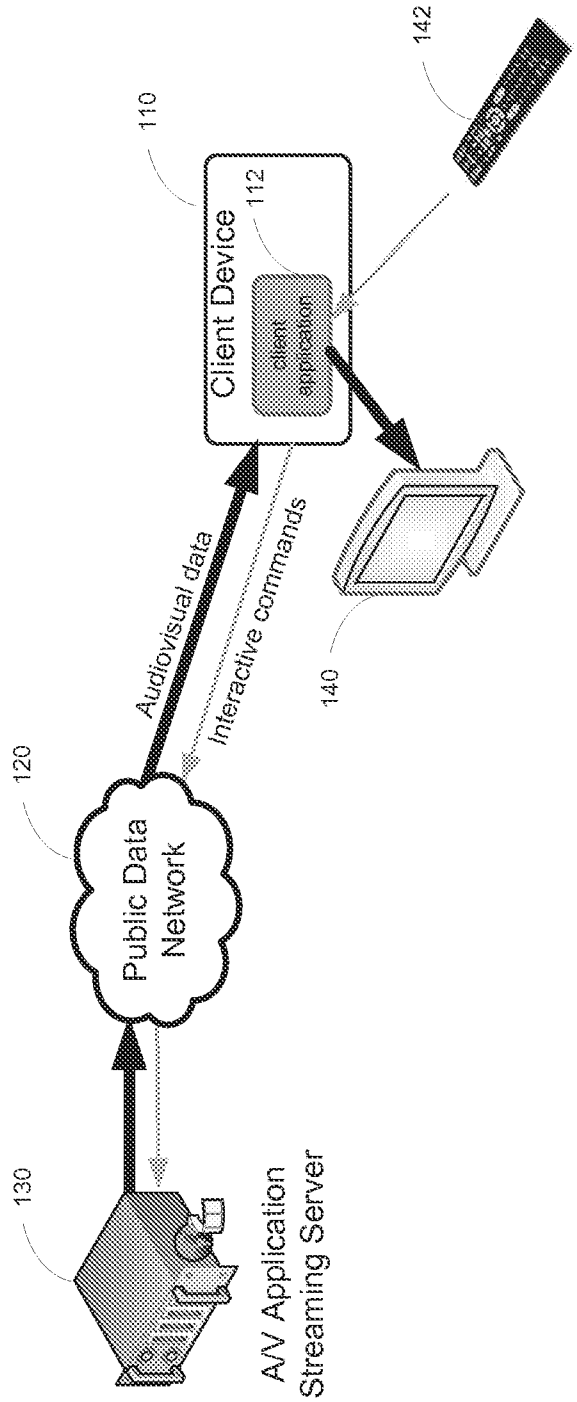
142
110
Client Device
112
Client application
Audiovisual data
Interactive commands
140
120
Public Data Network
130
A/V Application Streaming Server

FIG. 1

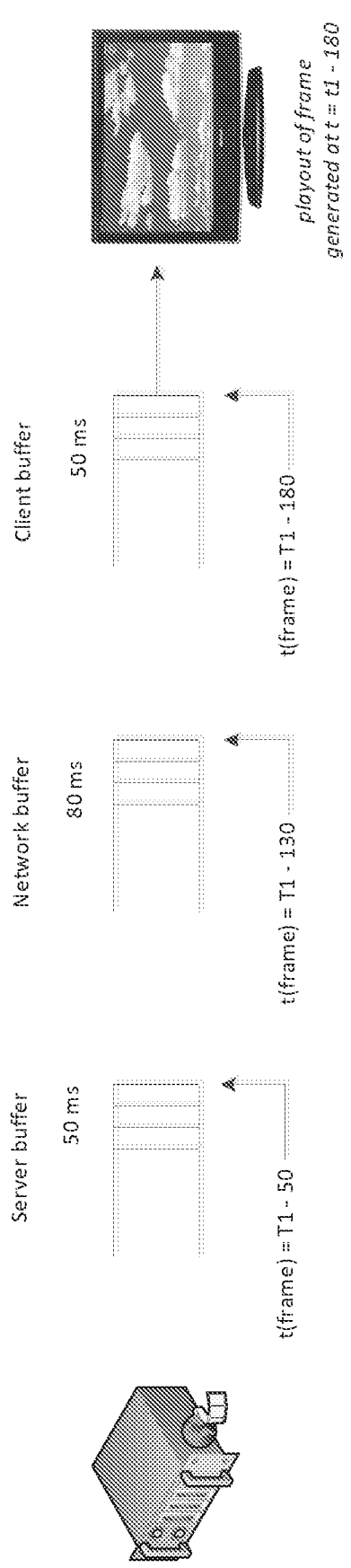
Server buffer
50 ms
t(frame) = T1 - 50
Network buffer
80 ms
t(frame) = T1 - 130
Client buffer
50 ms
t(frame) = T1 - 180
playout of frame generated at t = t1 - 180

FIG. 2

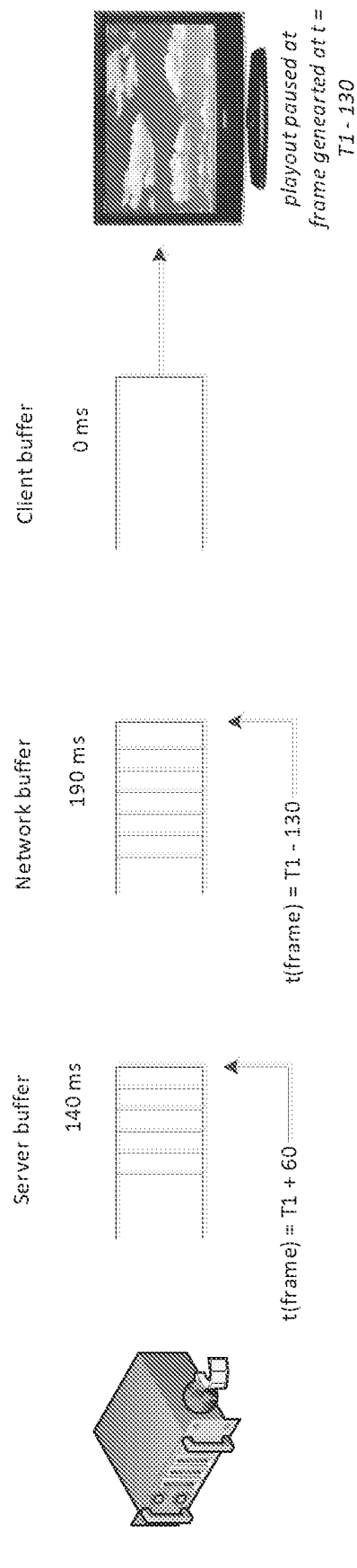
Server buffer
140 ms
t(frame) = T1 + 60
Network buffer
190 ms
t(frame) = T1 - 130
Client buffer
0 ms
playout paused at frame genearted at t = T1 - 130

FIG. 3

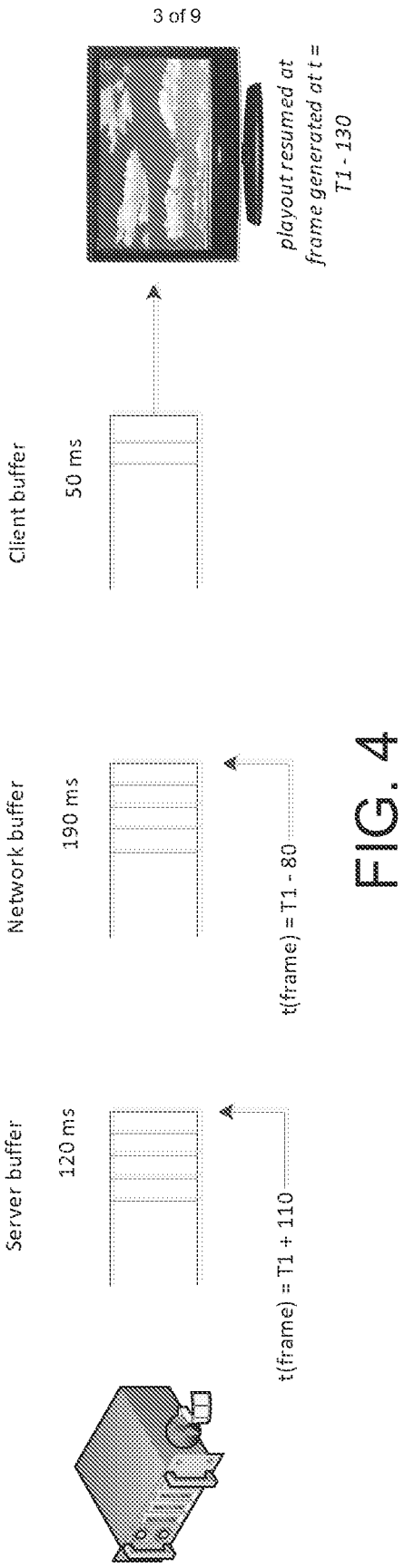
Server buffer
120 ms
t(frame) = T1 + 110
Network buffer
190 ms
t(frame) = T1 - 80
Client buffer
50 ms
playout resumed at frame generated at t = T1 - 130

FIG. 4

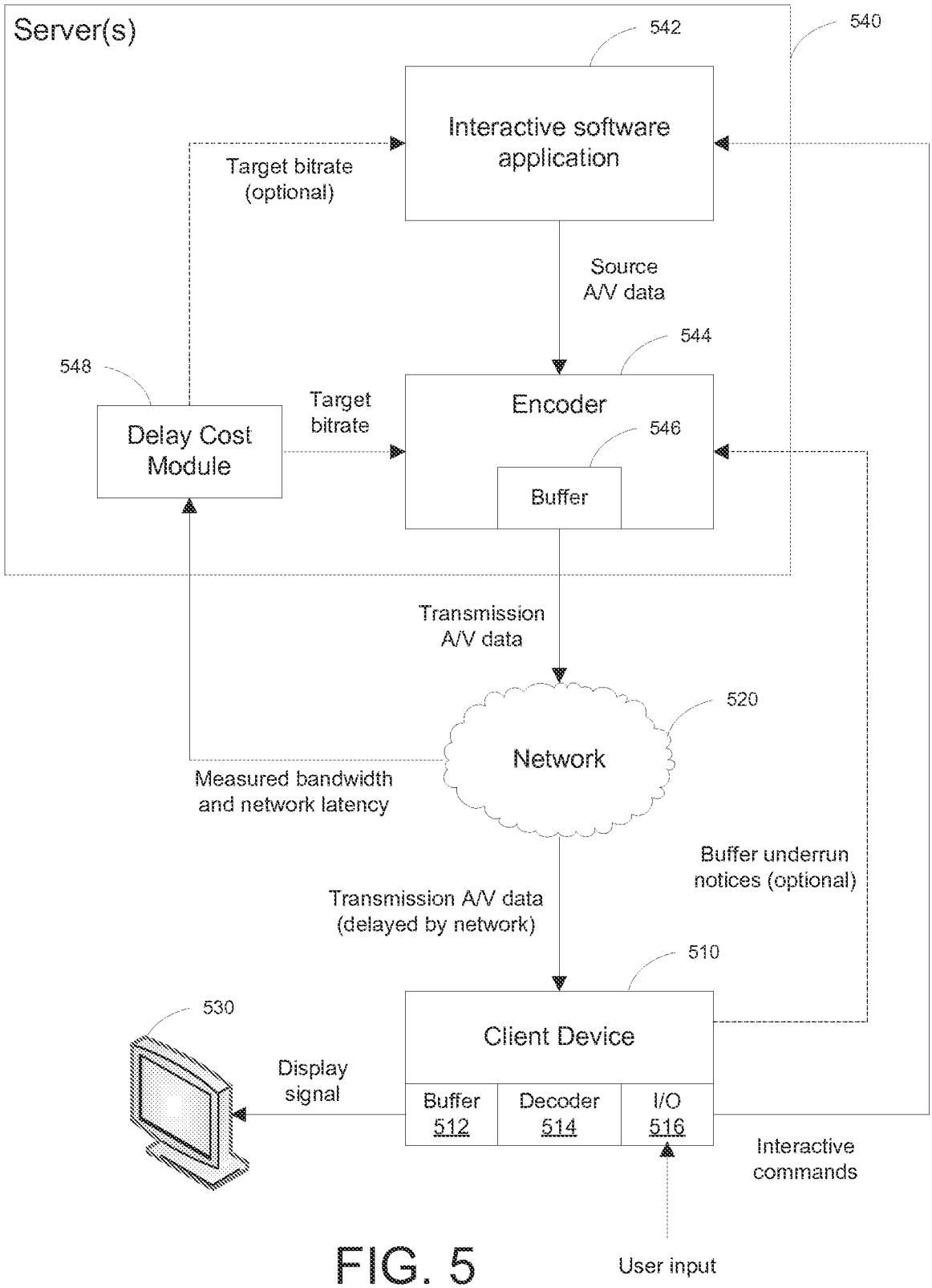
Server(s)
542
540
Interactive software application
Target bitrate (optional)
Source A/V data
544
548
Target bitrate
Encoder
Delay Cost Module
546
Buffer
Transmission A/V data
520
Network
Measured bandwidth and network latency
Buffer underrun notices (optional)
Transmission A/V data (delayed by network)
510
530
Client Device
Display signal
Buffer 512
Decoder 514
I/O 516
Interactive commands
User input

FIG. 5

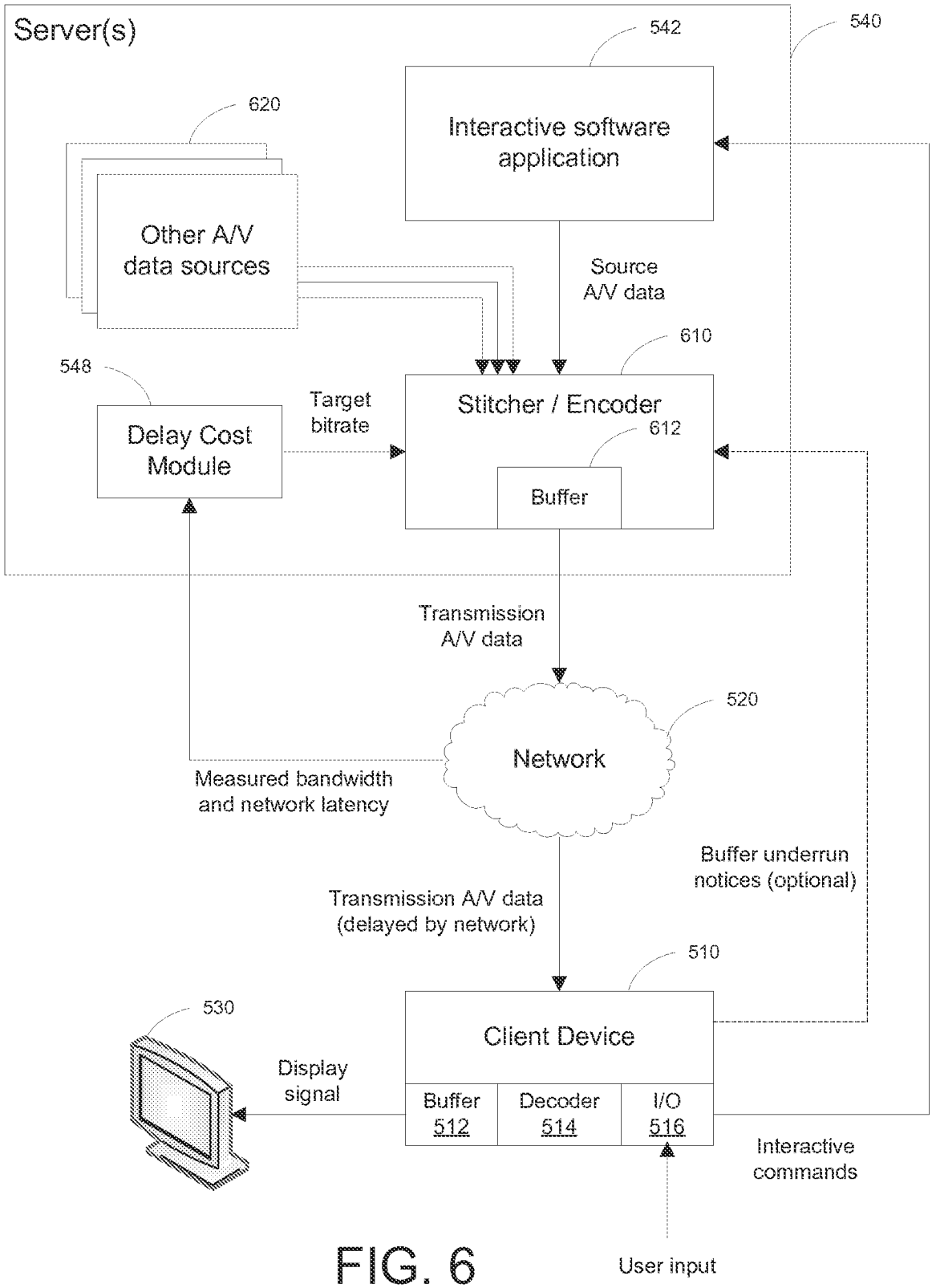
Server(s)
542
540
620
Interactive software application
Other A/V data sources
Source A/V data
610
548
Target bitrate
Stitcher / Encoder
Delay Cost Module
612
Buffer
Transmission A/V data
520
Network
Measured bandwidth and network latency
Buffer underrun notices (optional)
Transmission A/V data (delayed by network)
510
530
Client Device
Display signal
Buffer 512
Decoder 514
I/O 516
Interactive commands
User input

FIG. 6

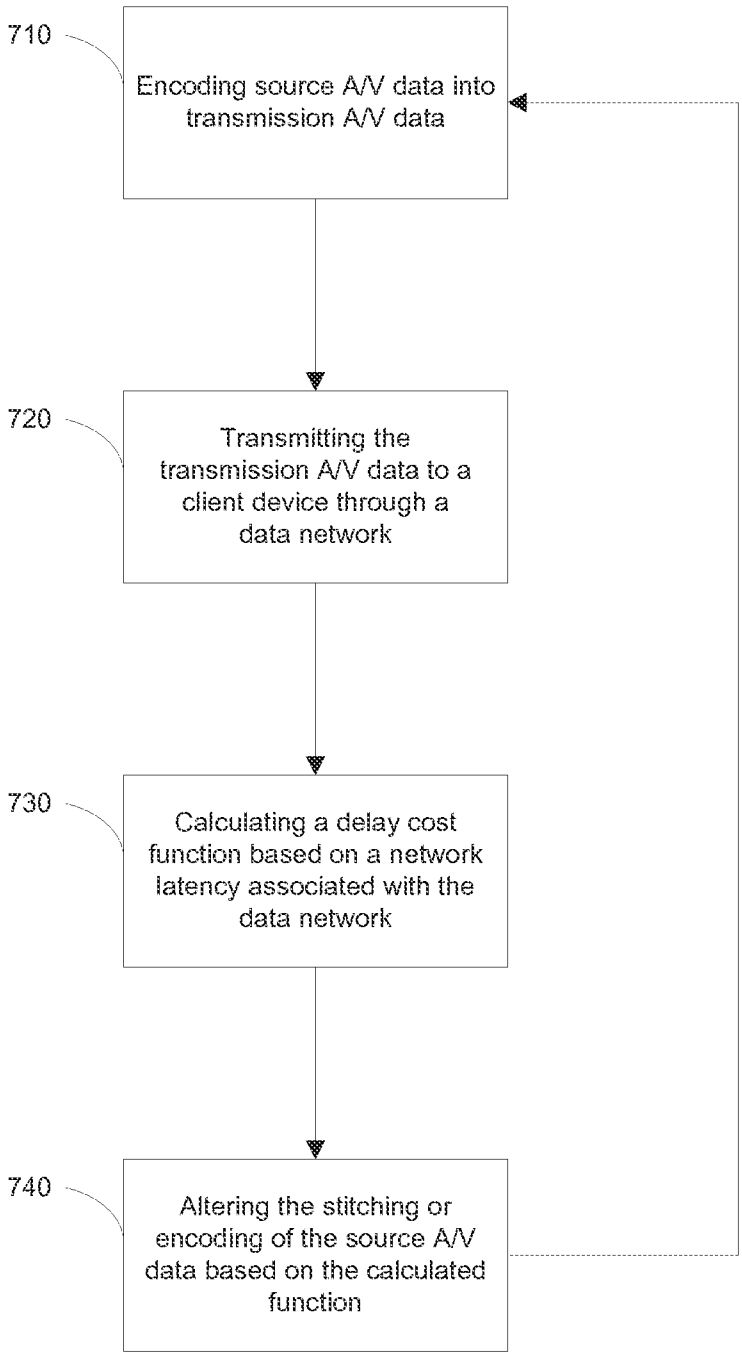
710
Encoding source A/V data into transmission A/V data
720
Transmitting the transmission A/V data to a client device through a data network
730
Calculating a delay cost function based on a network latency associated with the data network
740
Altering the stitching or encoding of the source A/V data based on the calculated function

FIG. 7

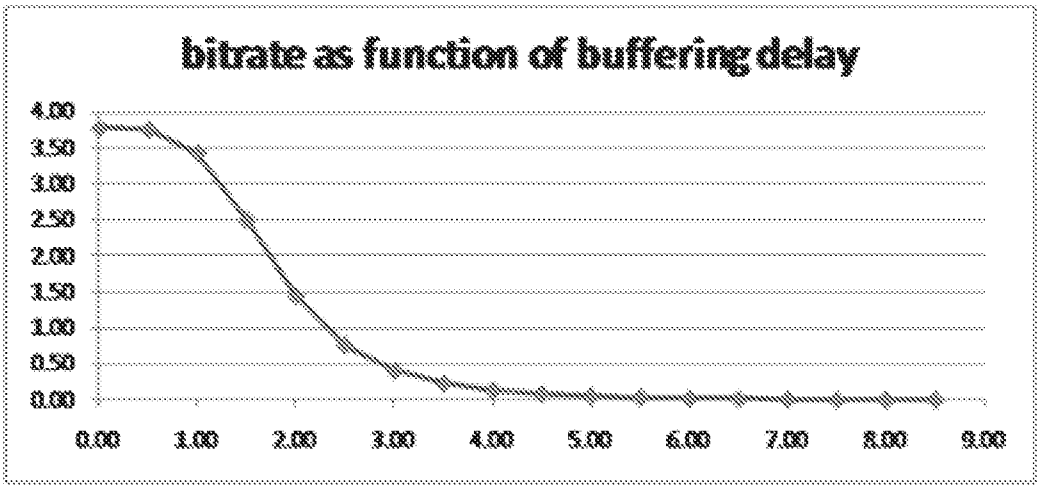
bitrate as function of buffering delay
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.00
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00

FIG. 8

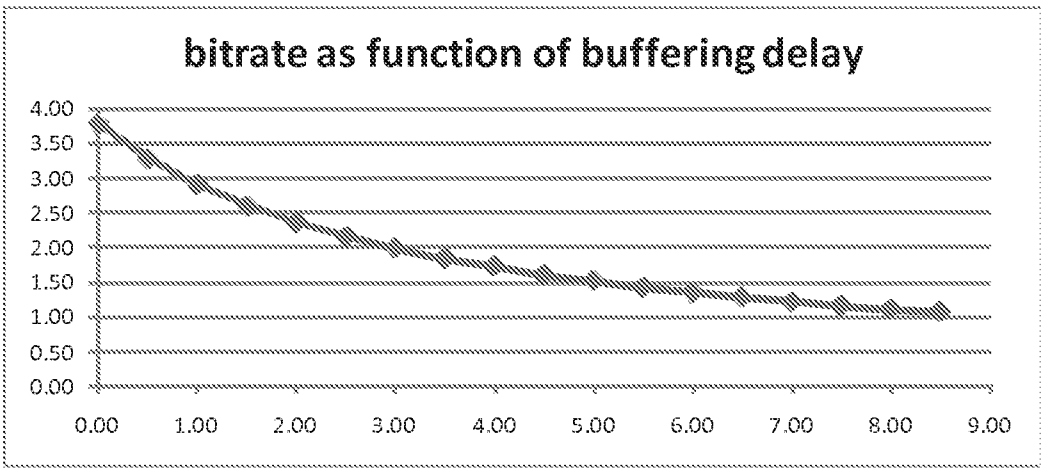
bitrate as function of buffering delay
4.00
3.50
3.00
2.50
2.00
1.50
1.00
0.50
0.00
0.00
1.00
2.00
3.00
4.00
5.00
6.00
7.00
8.00
9.00

FIG. 9

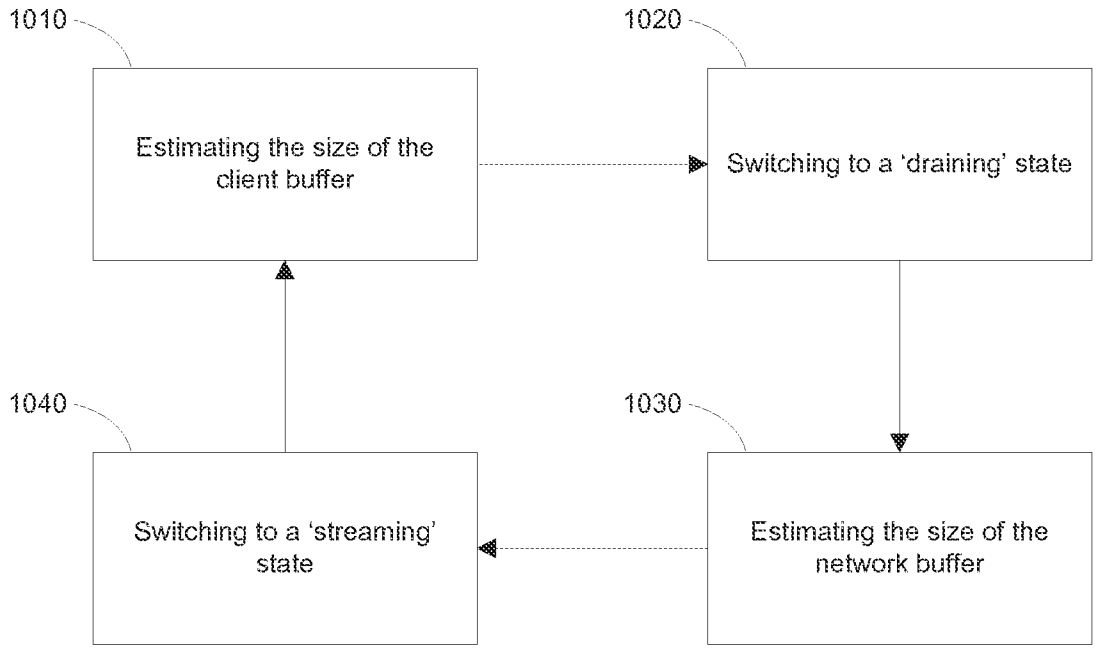
1010
Estimating the size of the client buffer
1020
Switching to a 'draining' state
1040
Switching to a 'streaming' state
1030
Estimating the size of the network buffer

FIG. 10

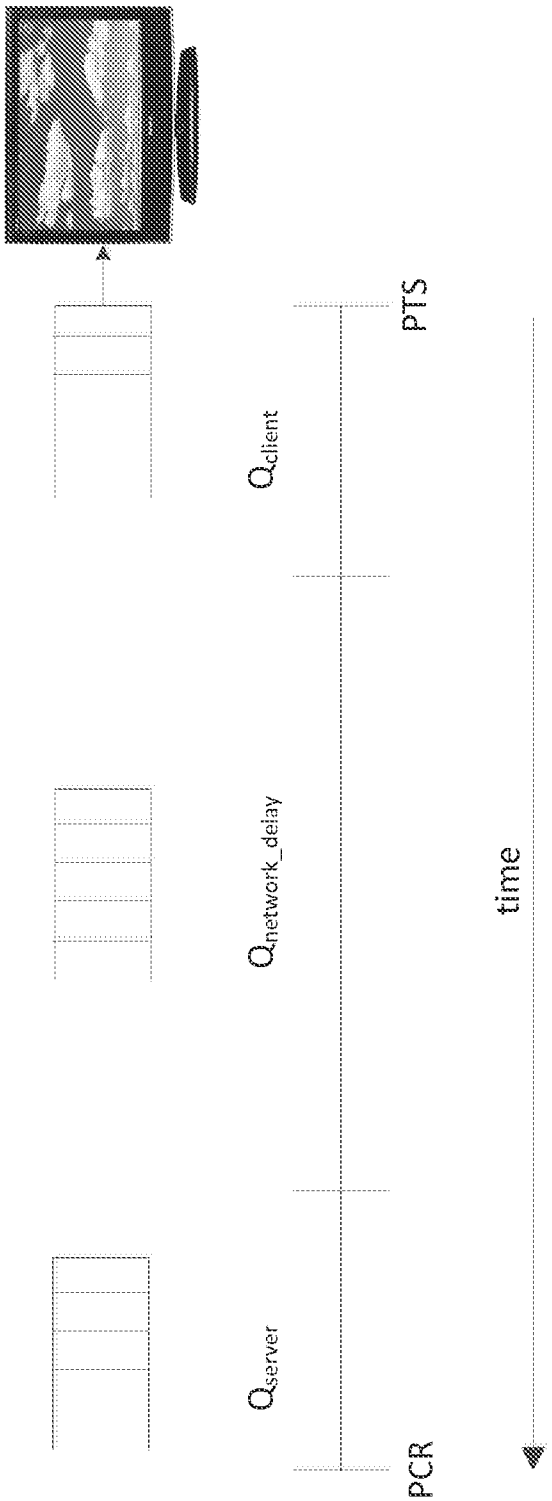

FIG. 11