US006292834B1

(12) **United States Patent**
Ravi et al.

(10) **Patent No.: US 6,292,834 B1**
(45) **Date of Patent: *Sep. 18, 2001**

(54) **DYNAMIC BANDWIDTH SELECTION FOR EFFICIENT TRANSMISSION OF MULTIMEDIA STREAMS IN A COMPUTER NETWORK**

(75) Inventors: **Hemanth Srinivas Ravi**, Milpitas; **Anders Edgar Klemets**; **Navin Chaddha**, both of Sunnyvale; **David de Val**, Mountain View, all of CA (US)

(73) Assignee: **Microsoft Corporation**, Redmond, WA (US)

(*) Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).
Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **08/818,127**

(22) Filed: **Mar. 14, 1997**

(51) **Int. Cl.**[7] **G06F 15/167**; G06F 15/173

(52) **U.S. Cl.** **709/233**; 709/216; 709/225

(58) **Field of Search** 370/232, 233, 370/234, 236; 709/203, 207, 212, 216, 217, 218, 219, 223, 224, 225, 231, 232, 233, 102, 104, 105; 707/10

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,862,264	8/1989	Wells et al.	358/138
4,931,950	6/1990	Isle et al.	364/513

(List continued on next page.)

OTHER PUBLICATIONS

Yuang, M.C. et al. "Dynamic Video Playout Smoothing Method for Multimedia Applications" IEEE ICC'96, 1996.*
Feng, Wu–Chi et al. "A Comparison of Bandwidth Smoothing Techniques for the Transmission of Prerecorded Compressed Video" 16th Joint Conference of IEEE CCS, 1997.*

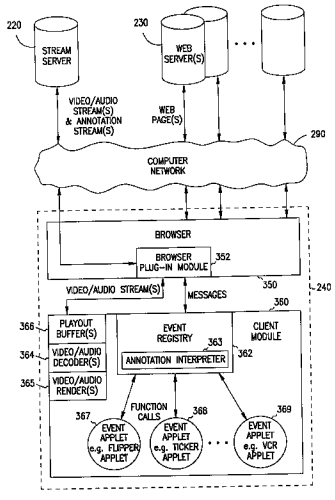
(List continued on next page.)

Primary Examiner—Zarni Maung
Assistant Examiner—Jason D. Cardone
(74) *Attorney, Agent, or Firm*—Lee & Hayes, PLLC

(57) **ABSTRACT**

An efficient transmission protocol for transmitting multimedia streams from a server to a client computer over a diverse computer network including local area networks (LANs) and wide area networks (WANs) such as the internet. The client computer includes a playout buffer, and the transmission rate is dynamically matched to the available bandwidth capacity of the network connection between the server and the client computer. If a playtime of the playout buffer, which is one measure of the number of data packets currently in the playout buffer, drops below a dynamically computed Decrease_Bandwidth (DEC_BW) threshold, then the transmission rate is decreased by sending a DEC_BW message to the server. Conversely, if the number of packets remaining in the playout buffer rises above a dynamically computed Upper Increase_Bandwidth (INC_BW) threshold and does not drop below a Lower INC_BW threshold for at least an INC_BW wait period, then the transmission rate is incremented. The transmission rate can be selected from among a predetermined set of discrete bandwidth values or from within a continuous range of bandwidth values. In one variation, in addition to responding to changes in network connection capacity, the client computer also determines an average client computational capacity. Accordingly, if the average client computational capacity is less than the network capacity, the lower of the two capacities is the determining one, thereby avoiding a playout buffer overrun.

42 Claims, 18 Drawing Sheets



U.S. PATENT DOCUMENTS

5,025,457	*	6/1991	Ahmed	375/354
5,050,161		9/1991	Golestani	370/60
5,088,107		2/1992	Piasecki et al.	375/10
5,119,474		6/1992	Beitel et al.	395/154
5,208,810	*	5/1993	Park	370/230
5,231,599		7/1993	Peters et al.	364/709.16
5,274,758		12/1993	Beitel et al.	395/154
5,313,454		5/1994	Bustini et al.	370/13
5,359,593	*	10/1994	Derby et al. .	
5,434,848		7/1995	Chimento, Jr. et al.	370/170
5,442,389		8/1995	Blahut et al.	348/7
5,455,910		10/1995	Johnson et al.	395/650
5,463,422	*	10/1995	Simpson et al.	348/390
5,467,413		11/1995	Barrett	382/236
5,485,211		1/1996	Kuzma	348/409
5,487,167		1/1996	Dinallo et al.	395/650
5,490,252		2/1996	Macera et al.	395/200.1
5,504,744		4/1996	Adams et al.	370/60.1
5,519,701		5/1996	Colmant et al.	370/60.1
5,524,193		6/1996	Covington et al.	395/154
5,533,021		7/1996	Branstad et al.	370/60.1
5,537,408		7/1996	Branstad et al.	370/79
5,543,850		8/1996	Pratt et al.	348/617
5,544,170	*	8/1996	Kasahara	370/253
5,566,175	*	10/1996	Davis	370/468
5,574,724		11/1996	Bales et al.	370/68.1
5,574,861	*	11/1996	Lorvig et al. .	
5,577,258		11/1996	Cruz et al.	395/800
5,583,980		12/1996	Anderson	395/173
5,594,911		1/1997	Cruz et al.	395/800
5,600,775		2/1997	King et al.	395/806
5,602,992		2/1997	Danneels	709/248
5,623,690		4/1997	Palmer et al.	395/806
5,633,810		5/1997	Mandal et al.	364/514
5,633,859	*	5/1997	Jain et al.	370/234
5,666,487		9/1997	Goodman et al.	395/200.76
5,675,732		10/1997	Majeti et al.	395/200.01
5,717,691	*	2/1998	Dighe et al. .	
5,774,668		6/1998	Choquier et al.	395/200.53
5,796,724	*	8/1998	Rajamani et al. .	
5,812,788		9/1998	Agarwal	395/200.77
5,815,505		9/1998	Mills	370/522
5,822,524	*	10/1998	Chen et al.	709/203
5,825,771	*	10/1998	Cohen et al. .	
5,852,565		12/1998	Demos	364/715.02
5,859,667		1/1999	Kondo et al.	348/414
5,886,995		3/1999	Arsenault et al.	370/477
5,892,549		4/1999	Feng	348/422
5,916,307		6/1999	Piskiel et al.	709/314
5,918,002	*	6/1999	Klemets et al.	395/182.16
5,926,226		7/1999	Proctor et al.	348/422
5,940,072		8/1999	Jahanghir et al.	345/327
5,956,088		9/1999	Shen et al.	348/385
5,978,544		11/1999	Shimada et al.	386/112
5,991,307		11/1999	Komuro et al.	370/473
5,995,650		11/1999	Migdal et al.	382/154
5,999,906		12/1999	Mercs et al.	704/500
6,012,100		1/2000	Frailong et al.	709/250

OTHER PUBLICATIONS

Rosado–Sosa, Carlos et al. "Jitter Compensation Scheduling Schemes for the Support of Real–Time Communictions" IEEE ICC'98, 1998.*

"Web Theater Product User Guide, Version 2.0", Palo Alto, CA: VXtreme, Inc., (1997).

Bolot, J.C., et al., "Scalable Feedback Control for Multicast Video Distribution in the Internet", Conference Proceedings, ACM SIGCOMM '94, London, England, 58–67, (1994).

Yavatkar, R., et al., "Optimistic Strategies for Large–Scale Dissemination of Multimedia Information", Conference Proceedings, Multimedia '93, 13–20, (1993).

* cited by examiner

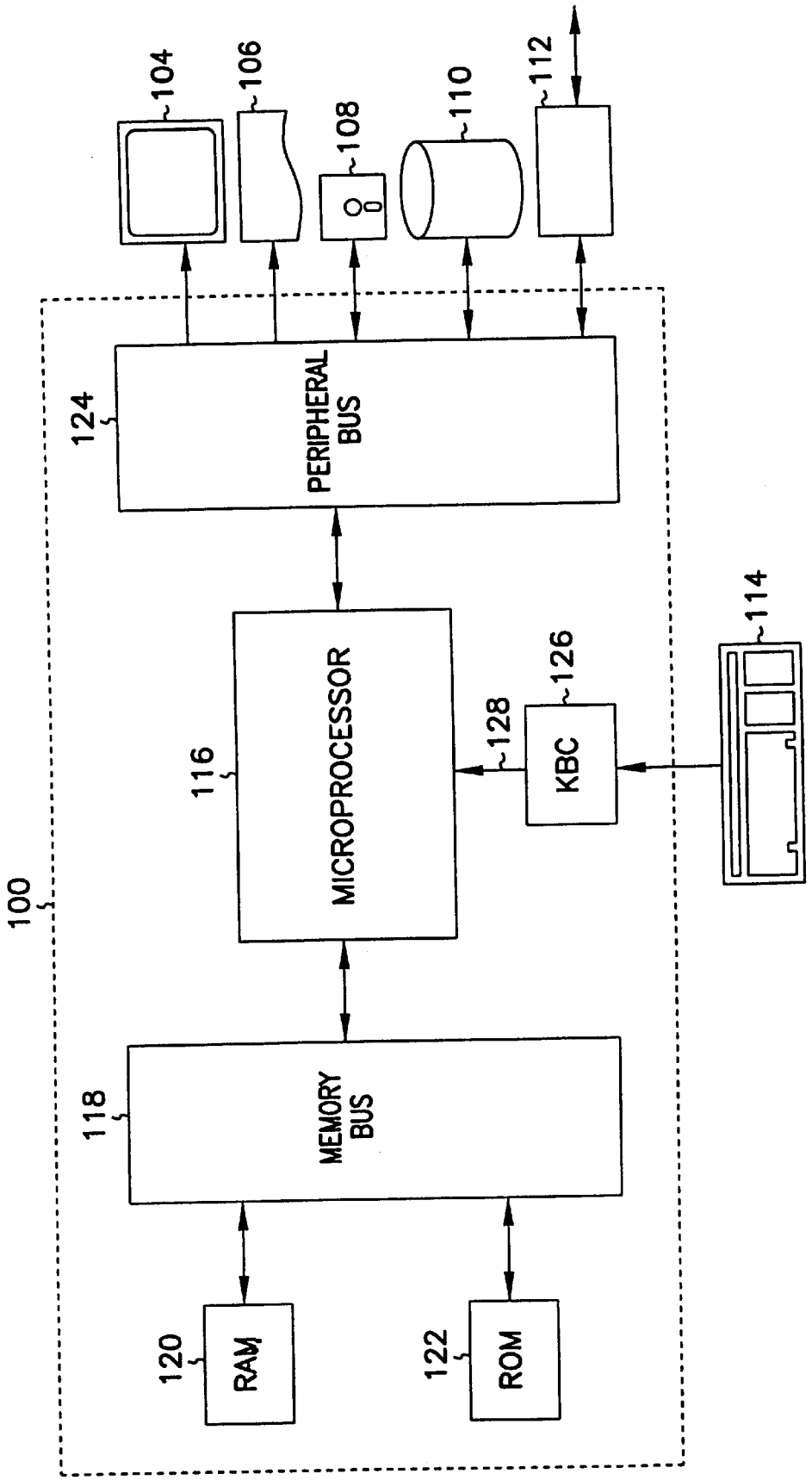

104
106
108
110
112
PERIPHERAL BUS
124
MICROPROCESSOR
116
KBC
126
128
114
100
MEMORY BUS
118
RAM
120
ROM
122

FIG. 1

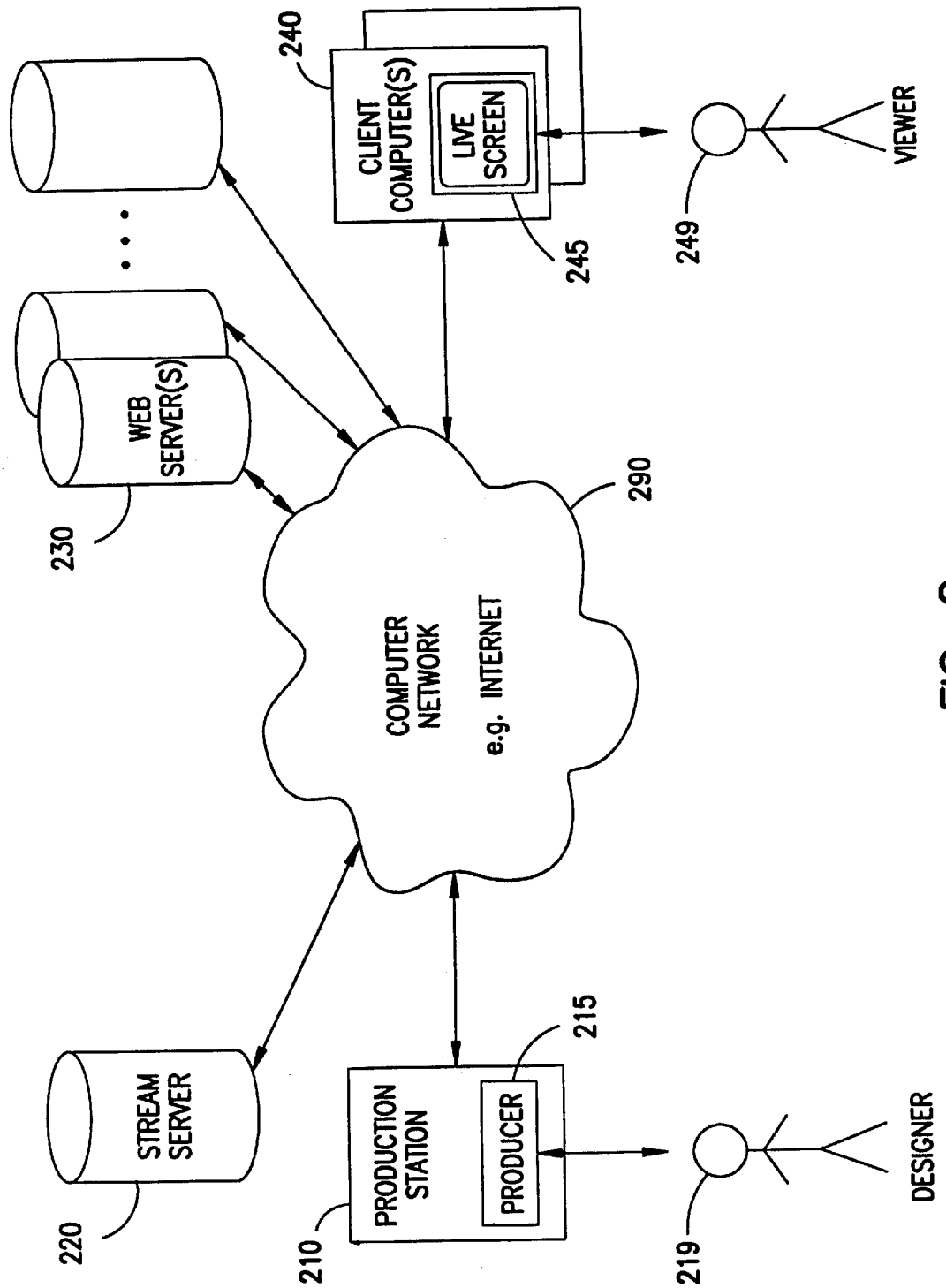

240
CLIENT COMPUTER(S)
LIVE SCREEN
245
249
VIEWER
WEB SERVER(S)
230
290
COMPUTER NETWORK
e.g. INTERNET
STREAM SERVER
220
PRODUCTION STATION
PRODUCER
215
210
219
DESIGNER

FIG. 2

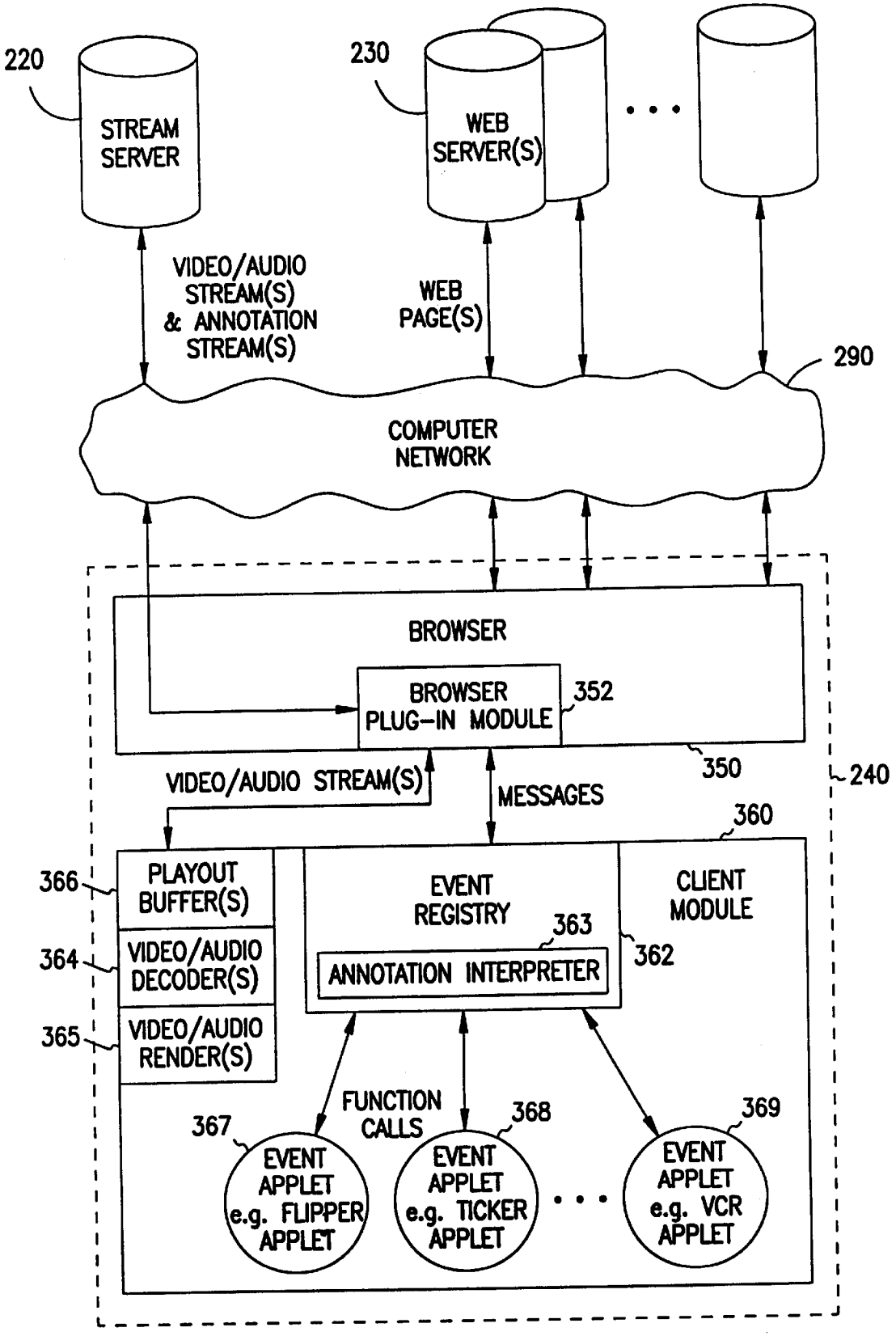

220
STREAM SERVER
230
WEB SERVER(S)
VIDEO/AUDIO STREAM(S) & ANNOTATION STREAM(S)
WEB PAGE(S)
290
COMPUTER NETWORK
BROWSER
BROWSER PLUG-IN MODULE
352
350
240
VIDEO/AUDIO STREAM(S)
MESSAGES
360
366
PLAYOUT BUFFER(S)
364
VIDEO/AUDIO DECODER(S)
365
VIDEO/AUDIO RENDER(S)
EVENT REGISTRY
363
ANNOTATION INTERPRETER
362
CLIENT MODULE
FUNCTION CALLS
367
EVENT APPLET e.g. FLIPPER APPLET
368
EVENT APPLET e.g. TICKER APPLET
369
EVENT APPLET e.g. VCR APPLET

FIG. 3

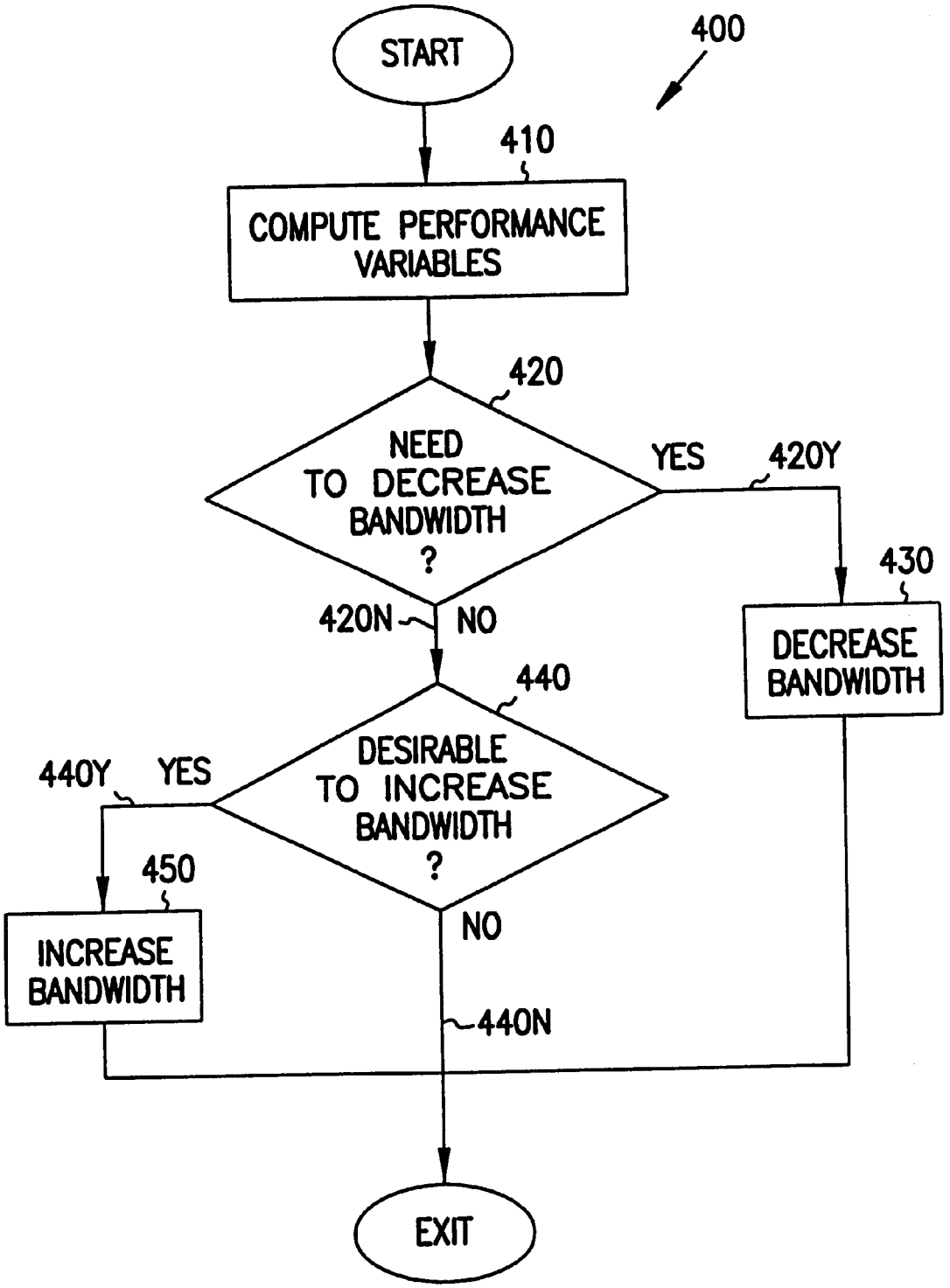

400
START
410
COMPUTE PERFORMANCE VARIABLES
420
NEED TO DECREASE BANDWIDTH ?
YES
420Y
430
DECREASE BANDWIDTH
420N
NO
440
DESIRABLE TO INCREASE BANDWIDTH ?
440Y
YES
450
INCREASE BANDWIDTH
NO
440N
EXIT

FIG. 4

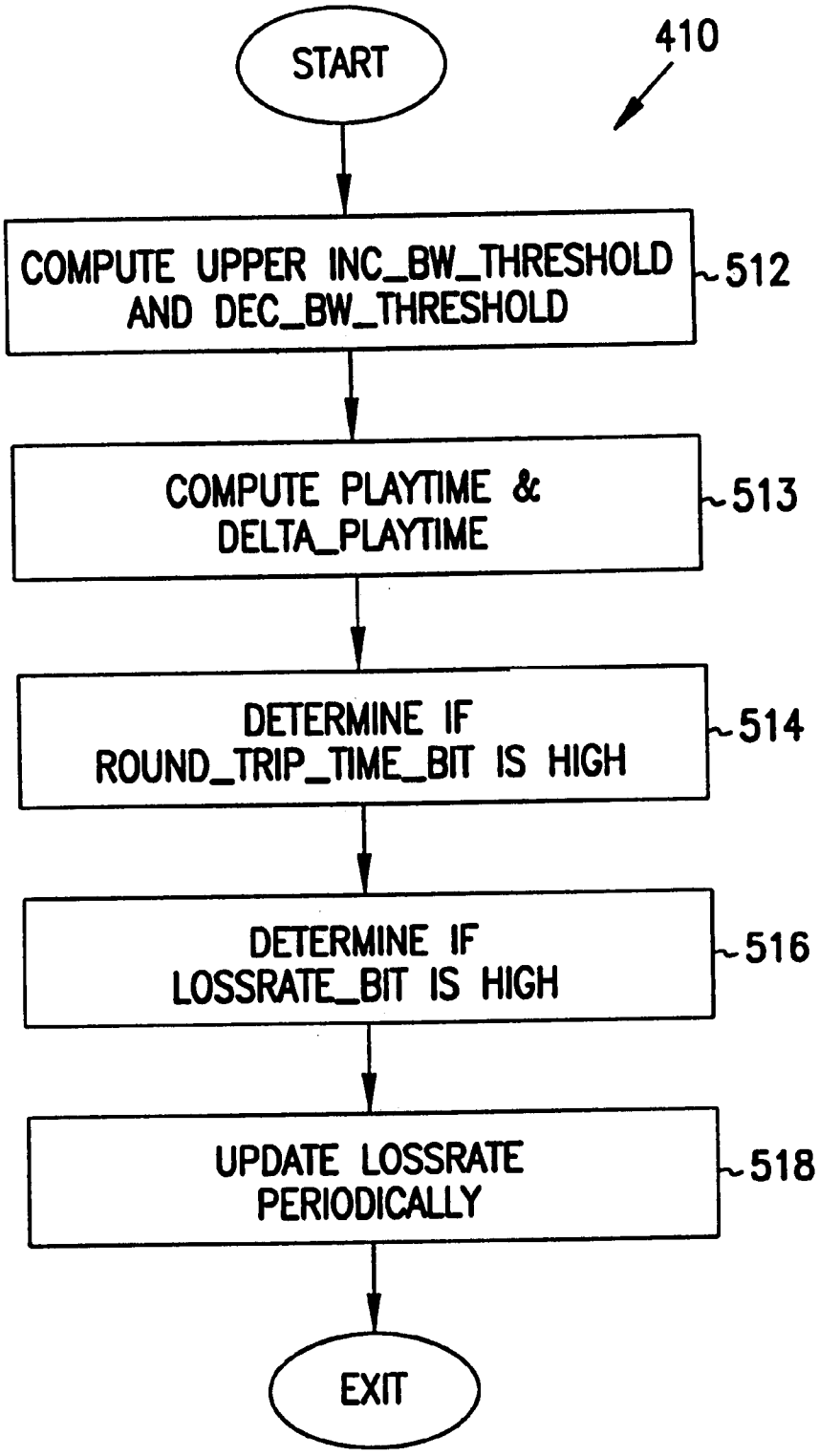

START
410
COMPUTE UPPER INC_BW_THRESHOLD AND DEC_BW_THRESHOLD
512
COMPUTE PLAYTIME & DELTA_PLAYTIME
513
DETERMINE IF ROUND_TRIP_TIME_BIT IS HIGH
514
DETERMINE IF LOSSRATE_BIT IS HIGH
516
UPDATE LOSSRATE PERIODICALLY
518
EXIT

FIG. 5A

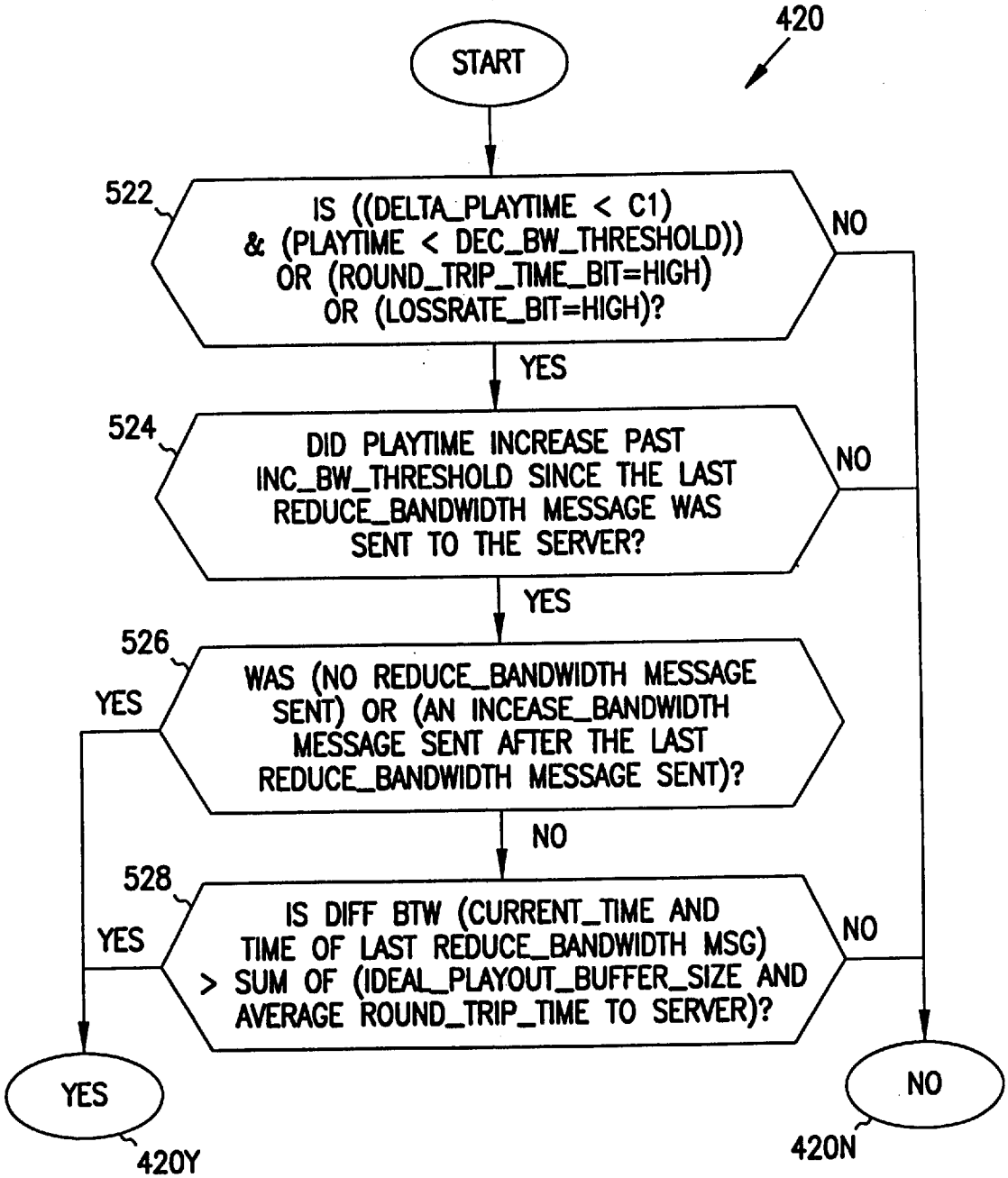

420
START
522
IS ((DELTA_PLAYTIME < C1)
& (PLAYTIME < DEC_BW_THRESHOLD))
OR (ROUND_TRIP_TIME_BIT=HIGH)
OR (LOSSRATE_BIT=HIGH)?
NO
YES
524
DID PLAYTIME INCREASE PAST
INC_BW_THRESHOLD SINCE THE LAST
REDUCE_BANDWIDTH MESSAGE WAS
SENT TO THE SERVER?
NO
YES
526
WAS (NO REDUCE_BANDWIDTH MESSAGE
SENT) OR (AN INCEASE_BANDWIDTH
MESSAGE SENT AFTER THE LAST
REDUCE_BANDWIDTH MESSAGE SENT)?
YES
NO
528
IS DIFF BTW (CURRENT_TIME AND
TIME OF LAST REDUCE_BANDWIDTH MSG)
> SUM OF (IDEAL_PLAYOUT_BUFFER_SIZE AND
AVERAGE ROUND_TRIP_TIME TO SERVER)?
YES
NO
YES
420Y
NO
420N

FIG. 5B

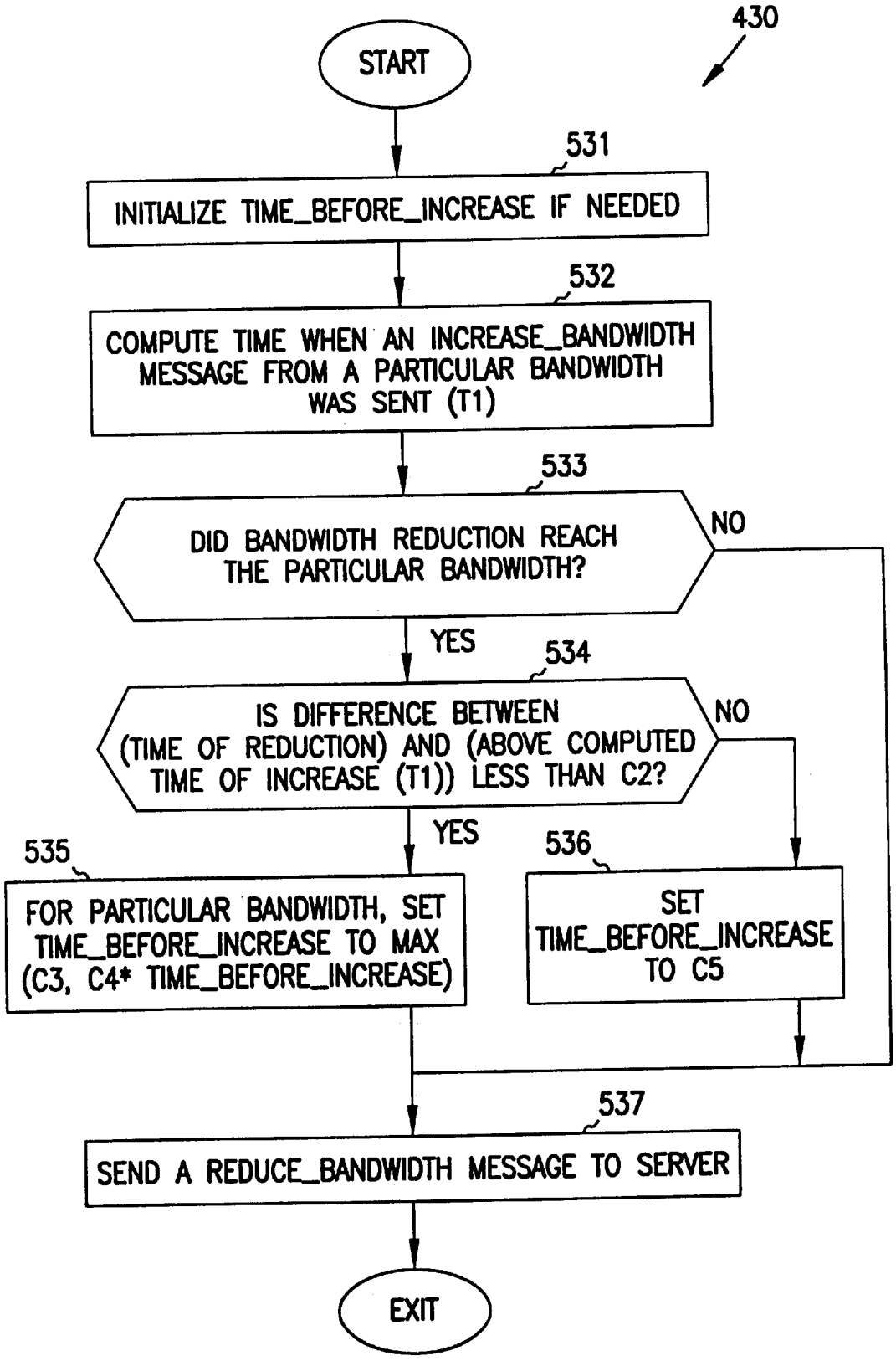

430
START
531
INITIALIZE TIME_BEFORE_INCREASE IF NEEDED
532
COMPUTE TIME WHEN AN INCREASE_BANDWIDTH MESSAGE FROM A PARTICULAR BANDWIDTH WAS SENT (T1)
533
DID BANDWIDTH REDUCTION REACH THE PARTICULAR BANDWIDTH?
NO
YES
534
IS DIFFERENCE BETWEEN (TIME OF REDUCTION) AND (ABOVE COMPUTED TIME OF INCREASE (T1)) LESS THAN C2?
NO
YES
535
FOR PARTICULAR BANDWIDTH, SET TIME_BEFORE_INCREASE TO MAX (C3, C4* TIME_BEFORE_INCREASE)
536
SET TIME_BEFORE_INCREASE TO C5
537
SEND A REDUCE_BANDWIDTH MESSAGE TO SERVER
EXIT

FIG. 5C

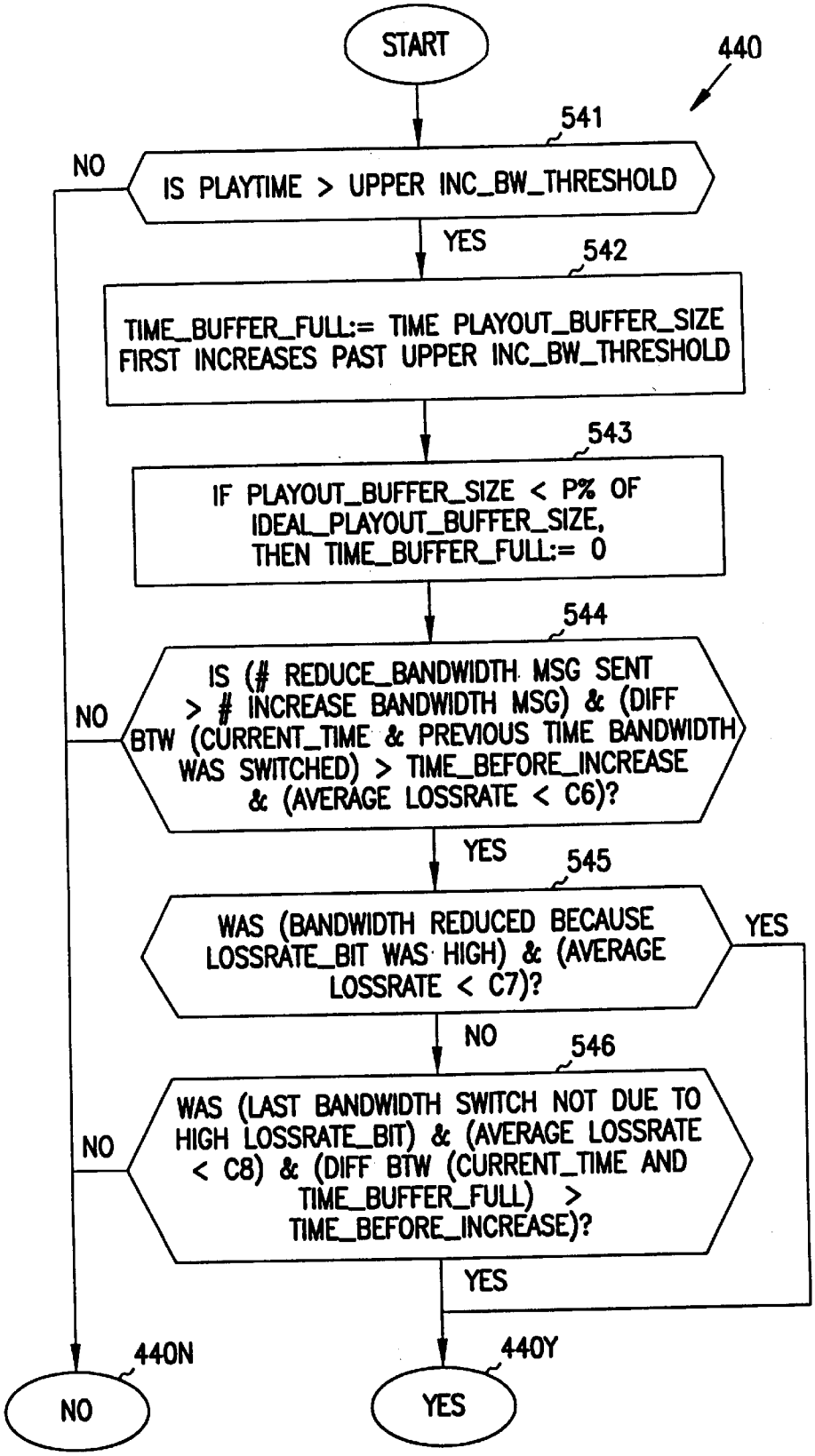

START
440
541
NO
IS PLAYTIME > UPPER INC_BW_THRESHOLD
YES
542
TIME_BUFFER_FULL:= TIME PLAYOUT_BUFFER_SIZE FIRST INCREASES PAST UPPER INC_BW_THRESHOLD
543
IF PLAYOUT_BUFFER_SIZE < P% OF IDEAL_PLAYOUT_BUFFER_SIZE, THEN TIME_BUFFER_FULL:= 0
544
NO
IS (# REDUCE_BANDWIDTH MSG SENT > # INCREASE BANDWIDTH MSG) & (DIFF BTW (CURRENT_TIME & PREVIOUS TIME BANDWIDTH WAS SWITCHED) > TIME_BEFORE_INCREASE & (AVERAGE LOSSRATE < C6)?
YES
545
WAS (BANDWIDTH REDUCED BECAUSE LOSSRATE_BIT WAS HIGH) & (AVERAGE LOSSRATE < C7)?
YES
NO
546
NO
WAS (LAST BANDWIDTH SWITCH NOT DUE TO HIGH LOSSRATE_BIT) & (AVERAGE LOSSRATE < C8) & (DIFF BTW (CURRENT_TIME AND TIME_BUFFER_FULL) > TIME_BEFORE_INCREASE)?
YES
440N
NO
440Y
YES

FIG. 5D

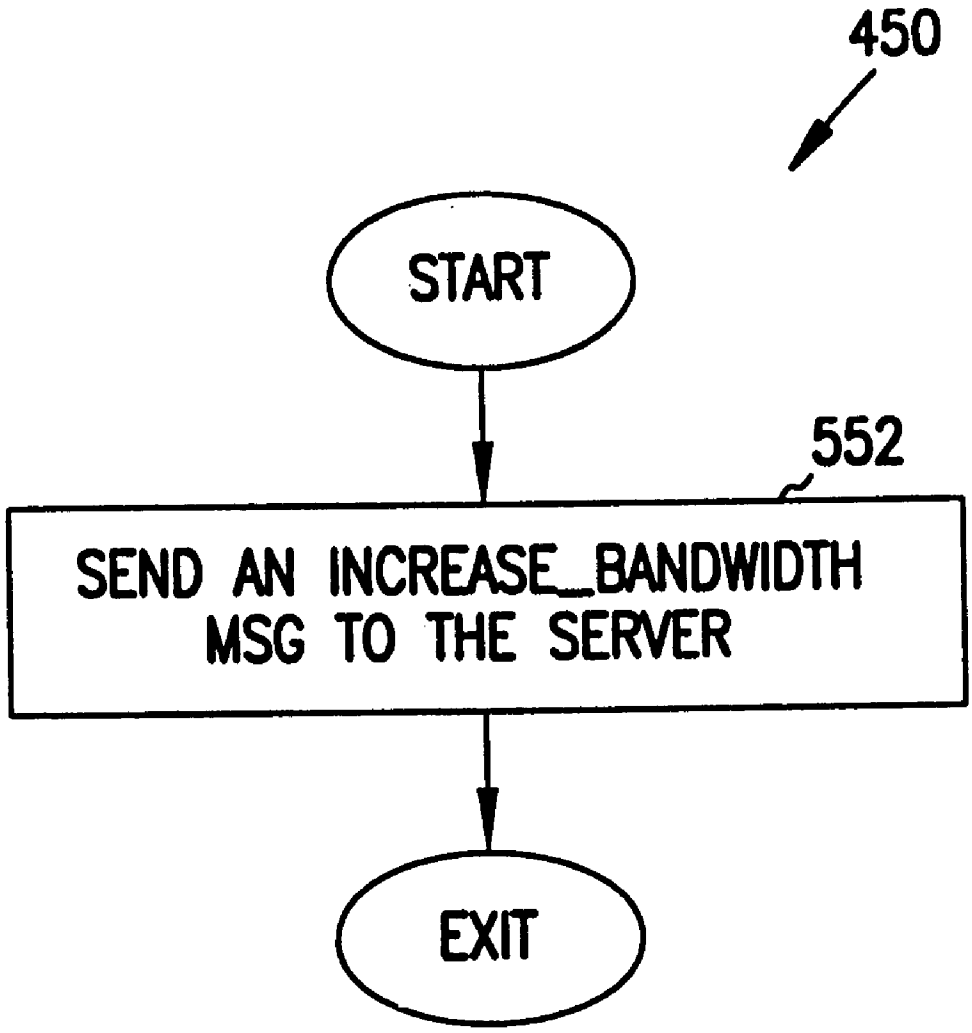

450
START
552
SEND AN INCREASE_BANDWIDTH MSG TO THE SERVER
EXIT

FIG. 5E

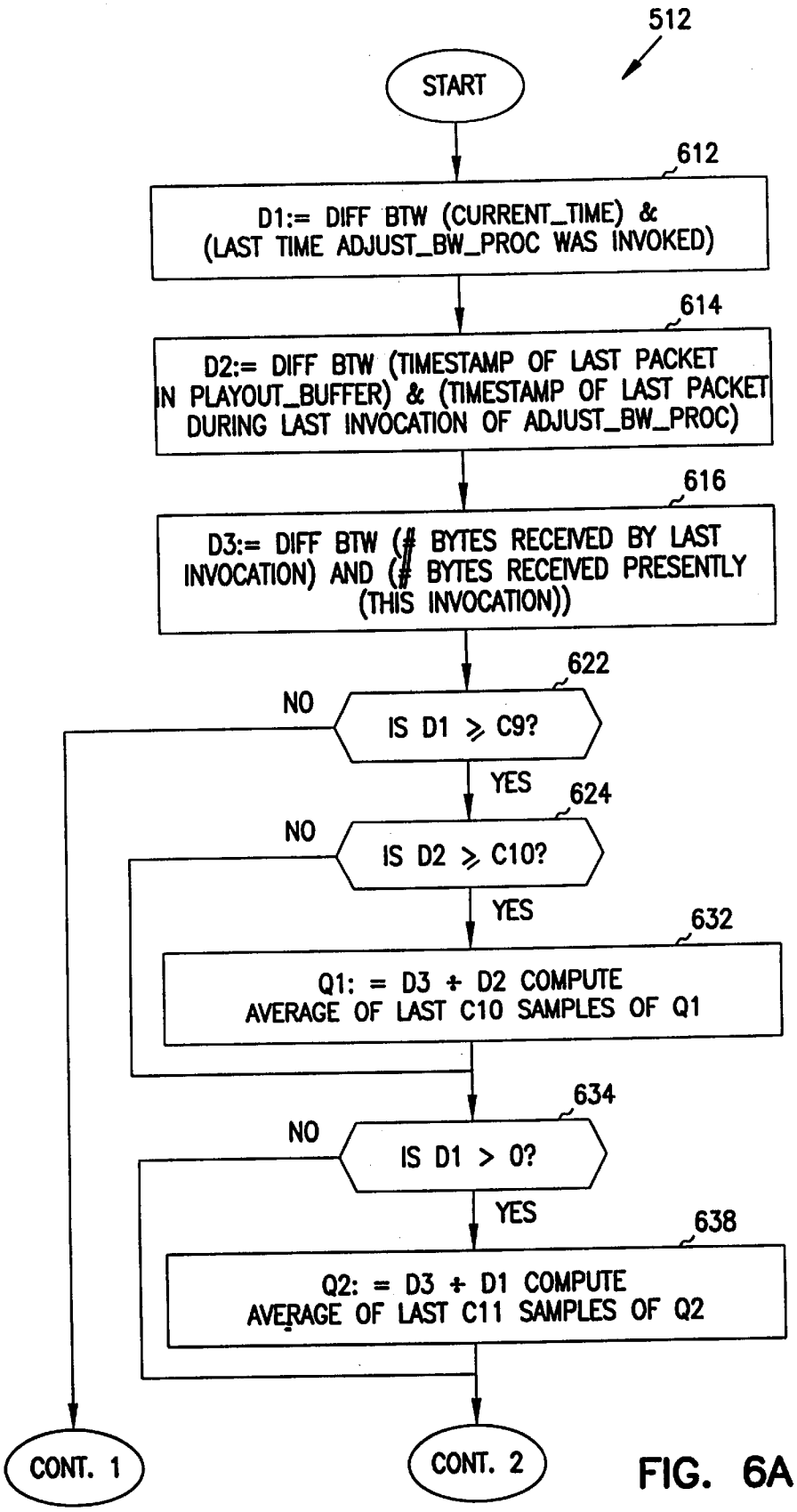

512
START
612
D1:= DIFF BTW (CURRENT_TIME) & (LAST TIME ADJUST_BW_PROC WAS INVOKED)
614
D2:= DIFF BTW (TIMESTAMP OF LAST PACKET IN PLAYOUT_BUFFER) & (TIMESTAMP OF LAST PACKET DURING LAST INVOCATION OF ADJUST_BW_PROC)
616
D3:= DIFF BTW (# BYTES RECEIVED BY LAST INVOCATION) AND (# BYTES RECEIVED PRESENTLY (THIS INVOCATION))
622
NO
IS D1 ≥ C9?
YES
624
NO
IS D2 ≥ C10?
YES
632
Q1: = D3 ÷ D2 COMPUTE AVERAGE OF LAST C10 SAMPLES OF Q1
634
NO
IS D1 > 0?
YES
638
Q2: = D3 ÷ D1 COMPUTE AVERAGE OF LAST C11 SAMPLES OF Q2
CONT. 1
CONT. 2

FIG. 6A

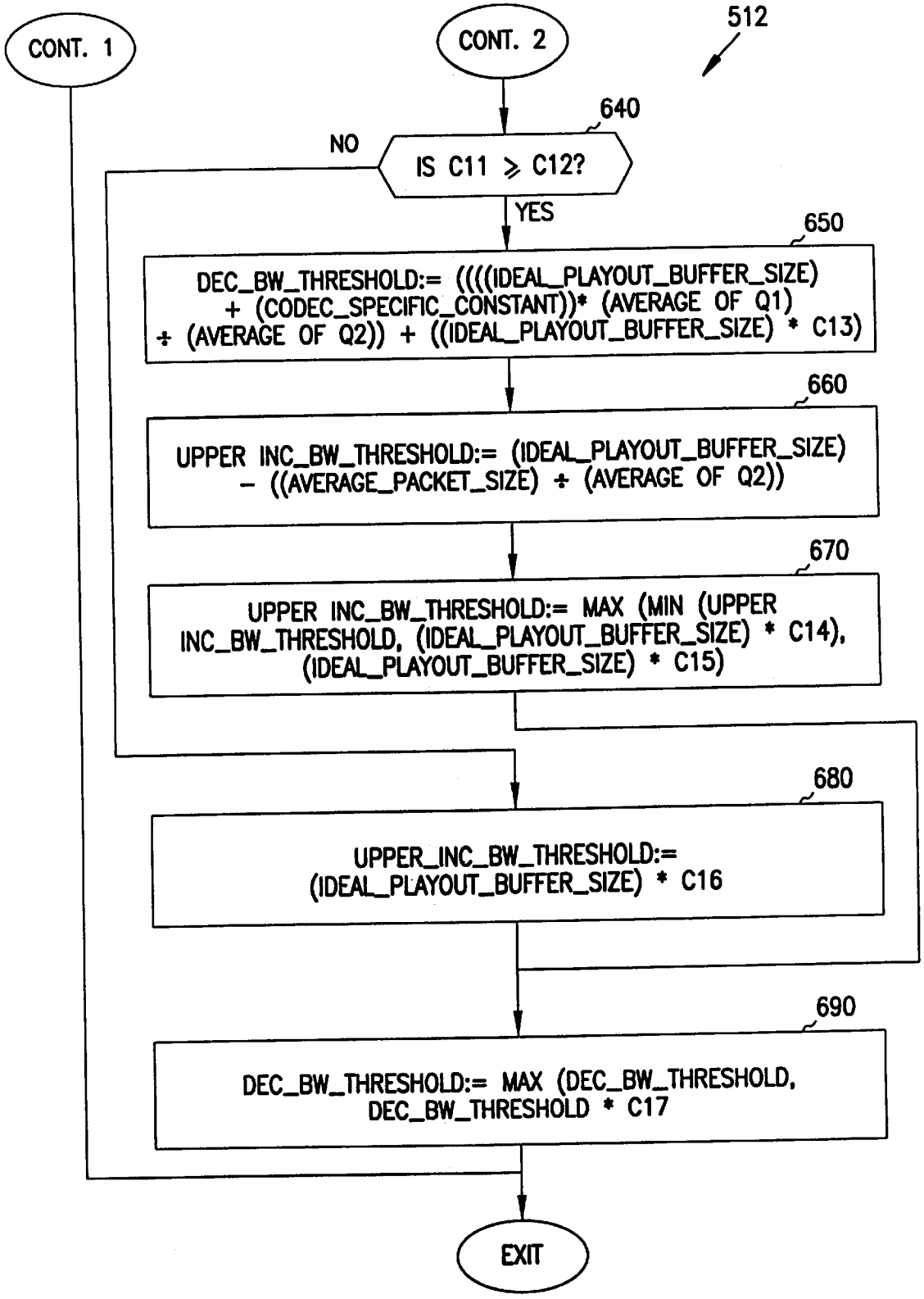

512
CONT. 1
CONT. 2
640
NO
IS C11 ≥ C12?
YES
650
DEC_BW_THRESHOLD:= ((((IDEAL_PLAYOUT_BUFFER_SIZE) + (CODEC_SPECIFIC_CONSTANT))* (AVERAGE OF Q1) ÷ (AVERAGE OF Q2)) + ((IDEAL_PLAYOUT_BUFFER_SIZE) * C13)
660
UPPER INC_BW_THRESHOLD:= (IDEAL_PLAYOUT_BUFFER_SIZE) − ((AVERAGE_PACKET_SIZE) ÷ (AVERAGE OF Q2))
670
UPPER INC_BW_THRESHOLD:= MAX (MIN (UPPER INC_BW_THRESHOLD, (IDEAL_PLAYOUT_BUFFER_SIZE) * C14), (IDEAL_PLAYOUT_BUFFER_SIZE) * C15)
680
UPPER_INC_BW_THRESHOLD:= (IDEAL_PLAYOUT_BUFFER_SIZE) * C16
690
DEC_BW_THRESHOLD:= MAX (DEC_BW_THRESHOLD, DEC_BW_THRESHOLD * C17
EXIT

FIG. 6B

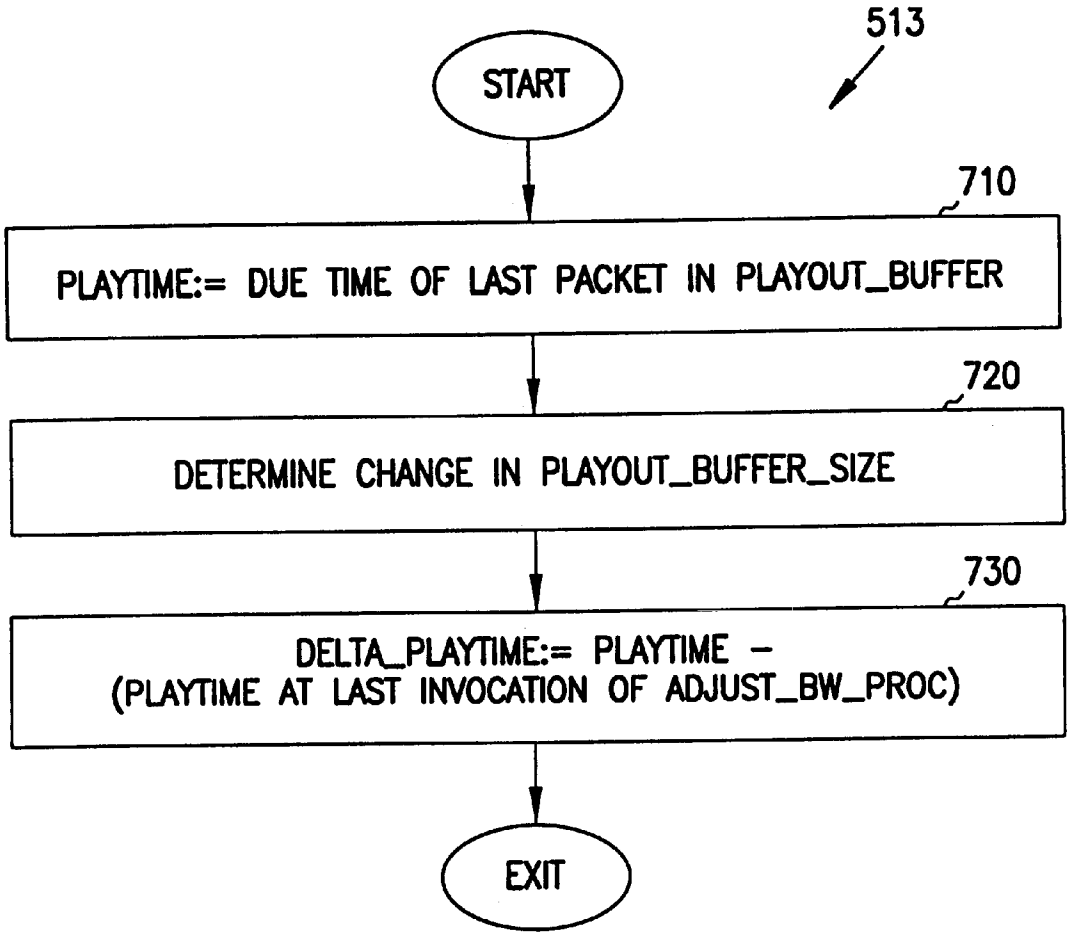

513
START
710
PLAYTIME:= DUE TIME OF LAST PACKET IN PLAYOUT_BUFFER
720
DETERMINE CHANGE IN PLAYOUT_BUFFER_SIZE
730
DELTA_PLAYTIME:= PLAYTIME -
(PLAYTIME AT LAST INVOCATION OF ADJUST_BW_PROC)
EXIT

FIG. 7A

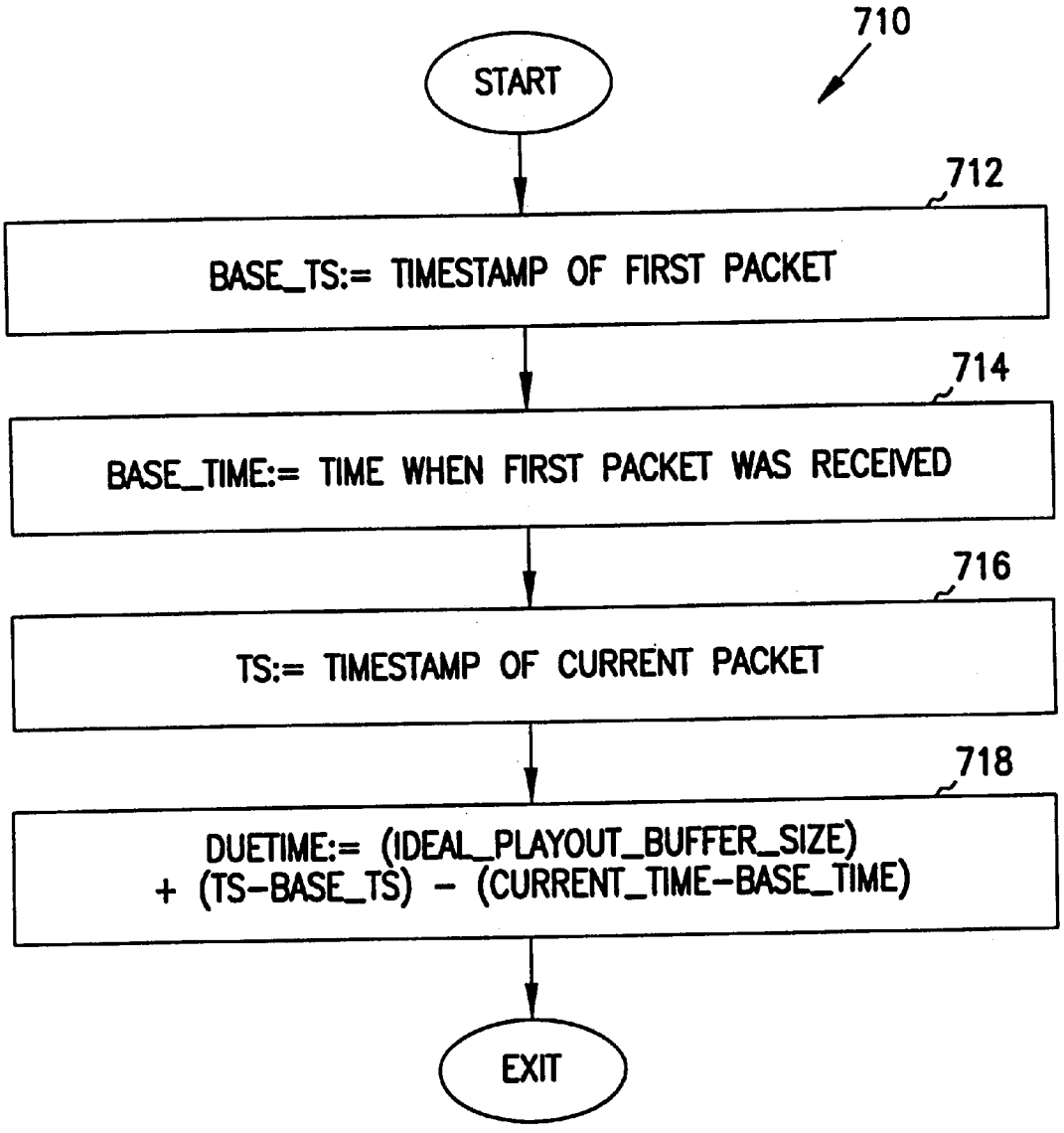

710
START
712
BASE_TS:= TIMESTAMP OF FIRST PACKET
714
BASE_TIME:= TIME WHEN FIRST PACKET WAS RECEIVED
716
TS:= TIMESTAMP OF CURRENT PACKET
718
DUETIME:= (IDEAL_PLAYOUT_BUFFER_SIZE)
+ (TS-BASE_TS) - (CURRENT_TIME-BASE_TIME)
EXIT

FIG. 7B

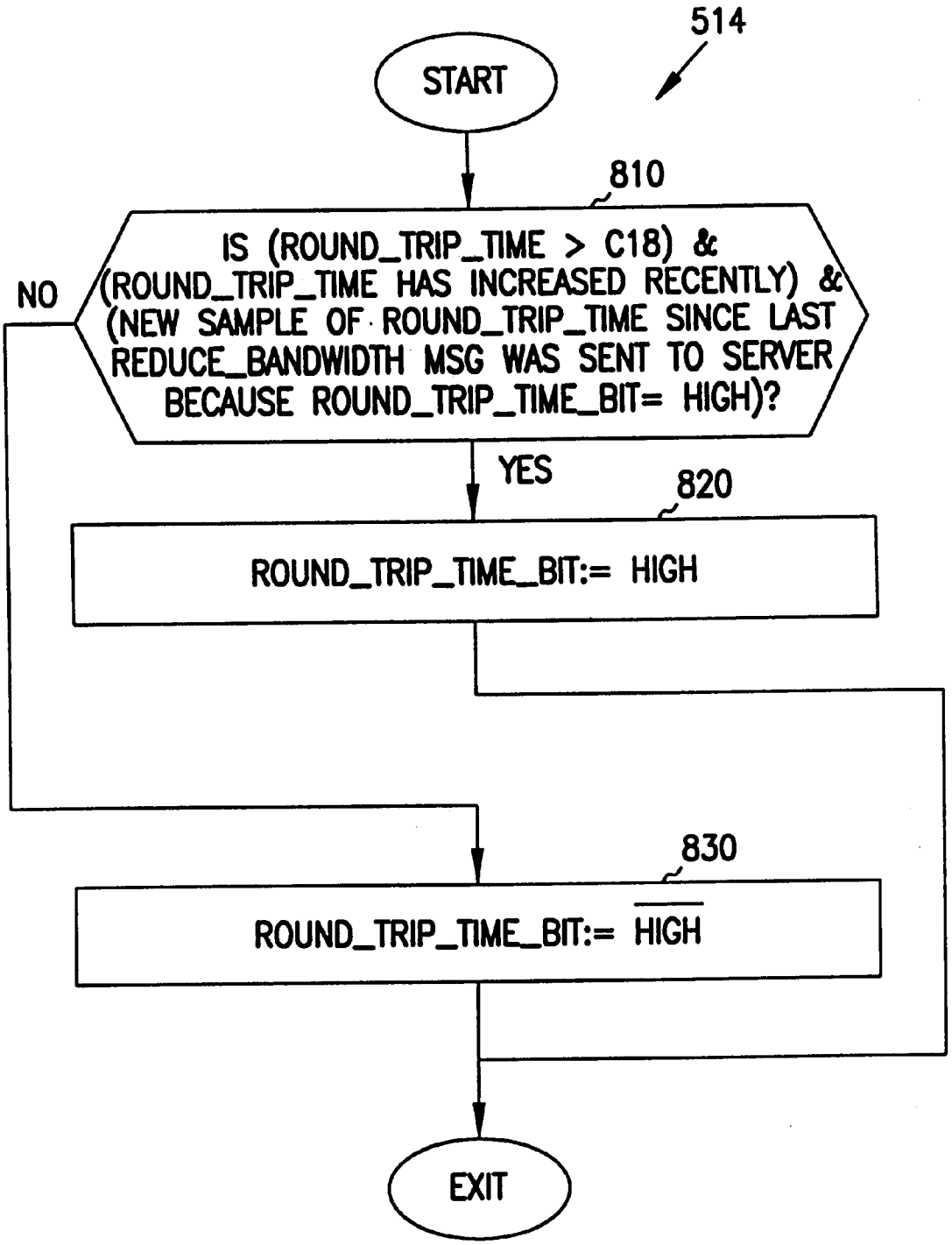

514
START
810
IS (ROUND_TRIP_TIME > C18) &
(ROUND_TRIP_TIME HAS INCREASED RECENTLY) &
(NEW SAMPLE OF ROUND_TRIP_TIME SINCE LAST
REDUCE_BANDWIDTH MSG WAS SENT TO SERVER
BECAUSE ROUND_TRIP_TIME_BIT= HIGH)?
NO
YES
820
ROUND_TRIP_TIME_BIT:= HIGH
830
ROUND_TRIP_TIME_BIT:= $\overline{\text{HIGH}}$
EXIT

FIG. 8

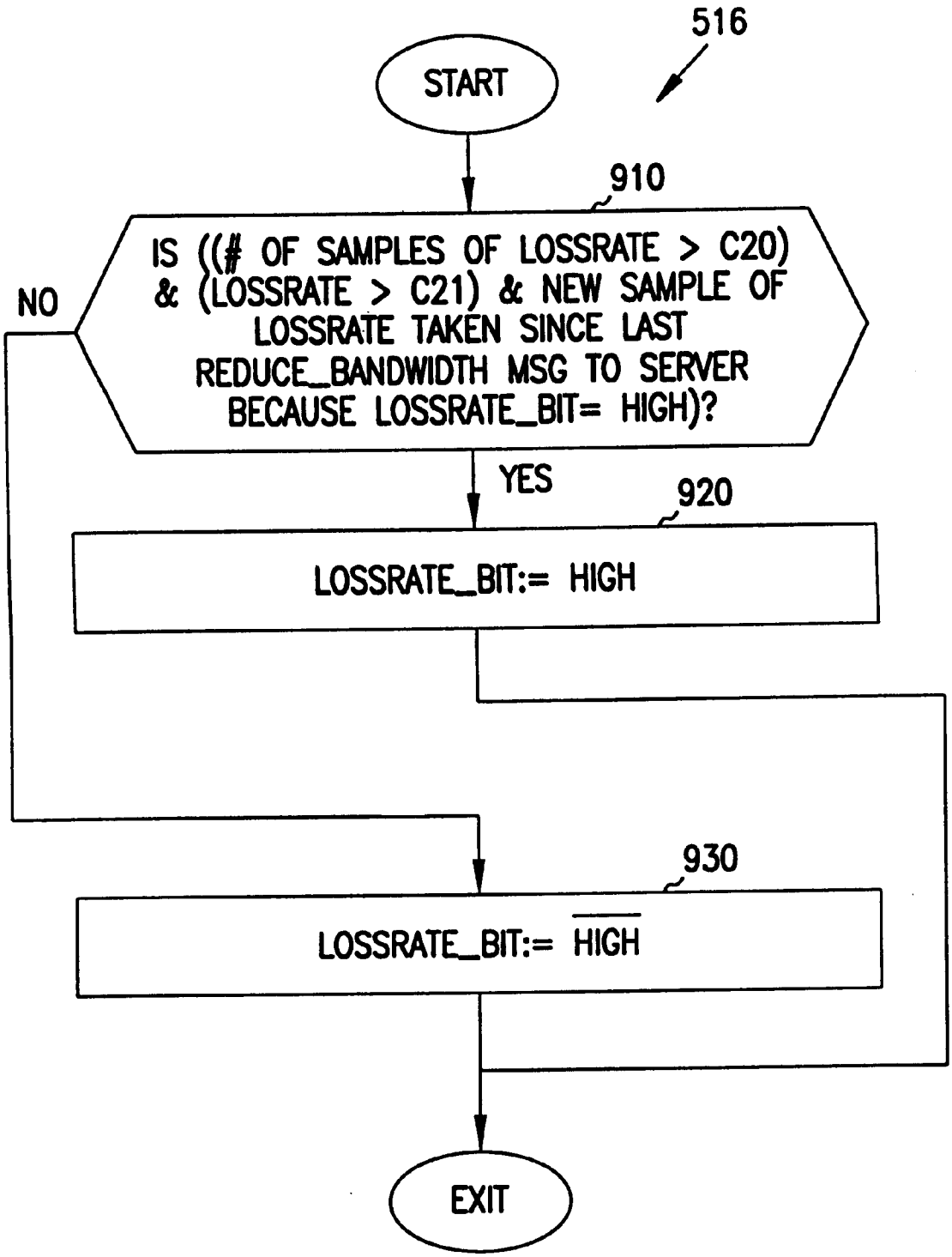

516
START
910
IS ((# OF SAMPLES OF LOSSRATE > C20) & (LOSSRATE > C21) & NEW SAMPLE OF LOSSRATE TAKEN SINCE LAST REDUCE_BANDWIDTH MSG TO SERVER BECAUSE LOSSRATE_BIT= HIGH)?
NO
YES
920
LOSSRATE_BIT:= HIGH
930
LOSSRATE_BIT:= $\overline{\text{HIGH}}$
EXIT

FIG. 9

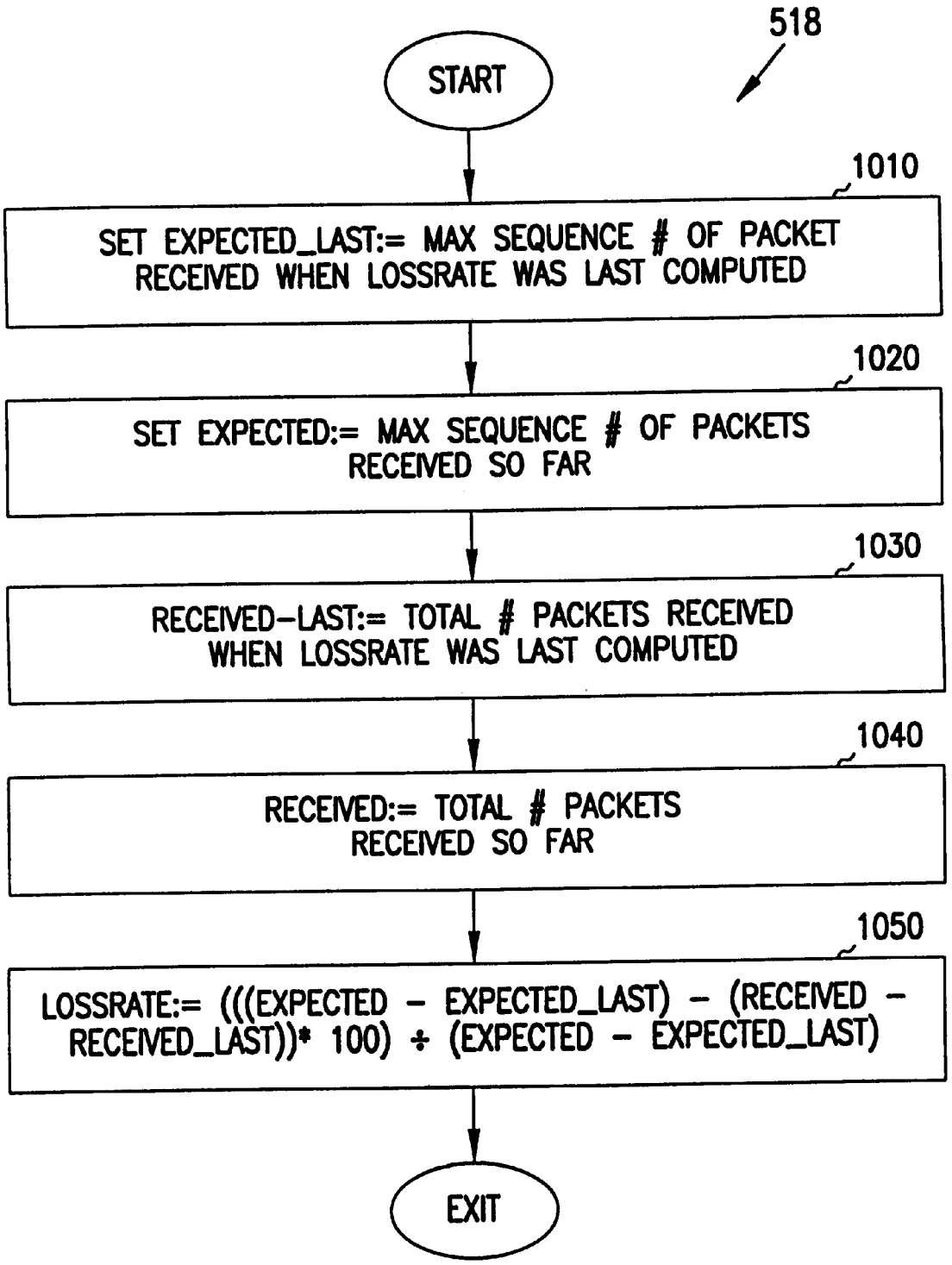

518
START
1010
SET EXPECTED_LAST:= MAX SEQUENCE # OF PACKET RECEIVED WHEN LOSSRATE WAS LAST COMPUTED
1020
SET EXPECTED:= MAX SEQUENCE # OF PACKETS RECEIVED SO FAR
1030
RECEIVED-LAST:= TOTAL # PACKETS RECEIVED WHEN LOSSRATE WAS LAST COMPUTED
1040
RECEIVED:= TOTAL # PACKETS RECEIVED SO FAR
1050
LOSSRATE:= (((EXPECTED - EXPECTED_LAST) - (RECEIVED - RECEIVED_LAST))* 100) ÷ (EXPECTED - EXPECTED_LAST)
EXIT

FIG. 10

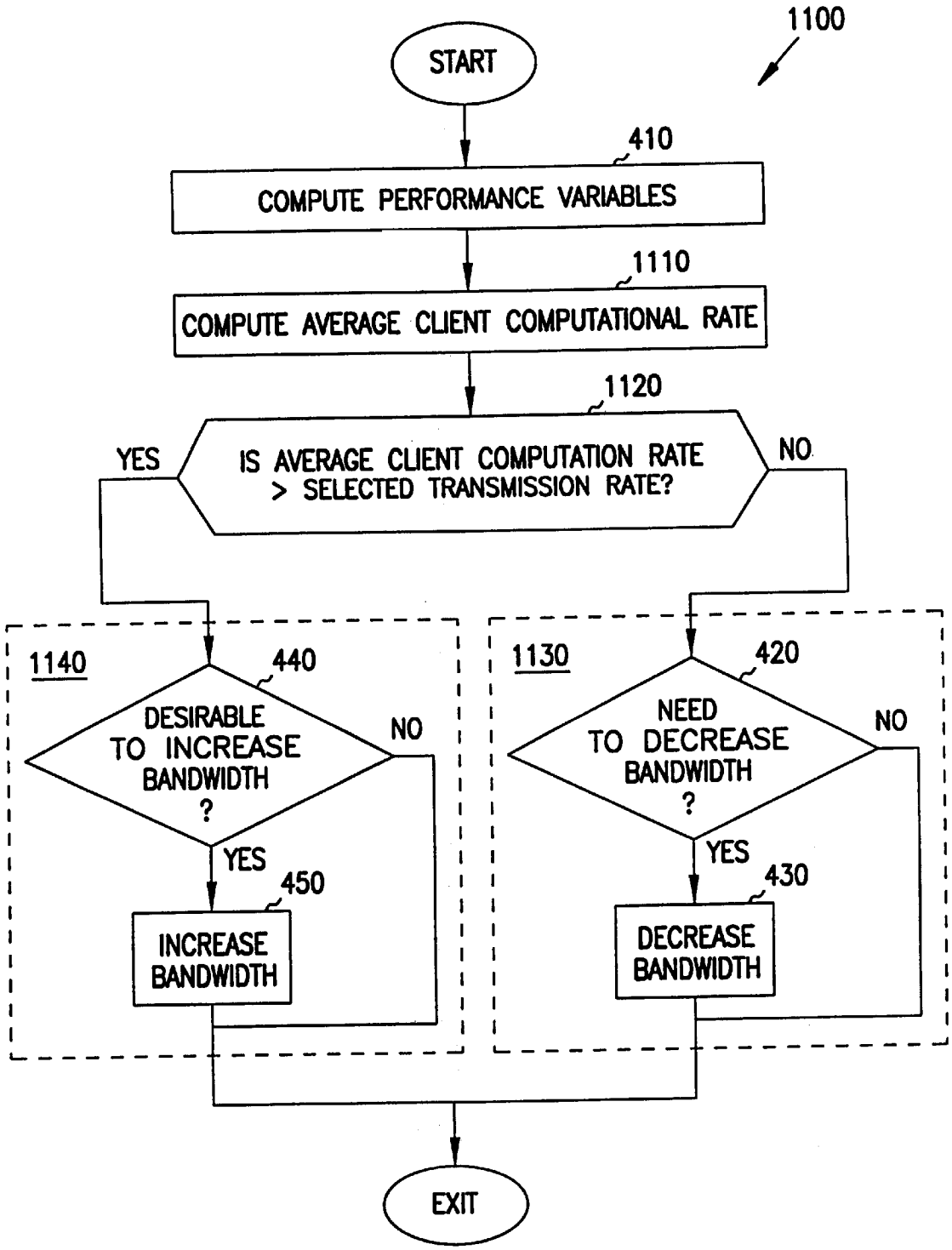

1100
START
410
COMPUTE PERFORMANCE VARIABLES
1110
COMPUTE AVERAGE CLIENT COMPUTATIONAL RATE
1120
IS AVERAGE CLIENT COMPUTATION RATE > SELECTED TRANSMISSION RATE?
YES
NO
1140
440
DESIRABLE TO INCREASE BANDWIDTH ?
NO
YES
450
INCREASE BANDWIDTH
1130
420
NEED TO DECREASE BANDWIDTH ?
NO
YES
430
DECREASE BANDWIDTH
EXIT

FIG. 11

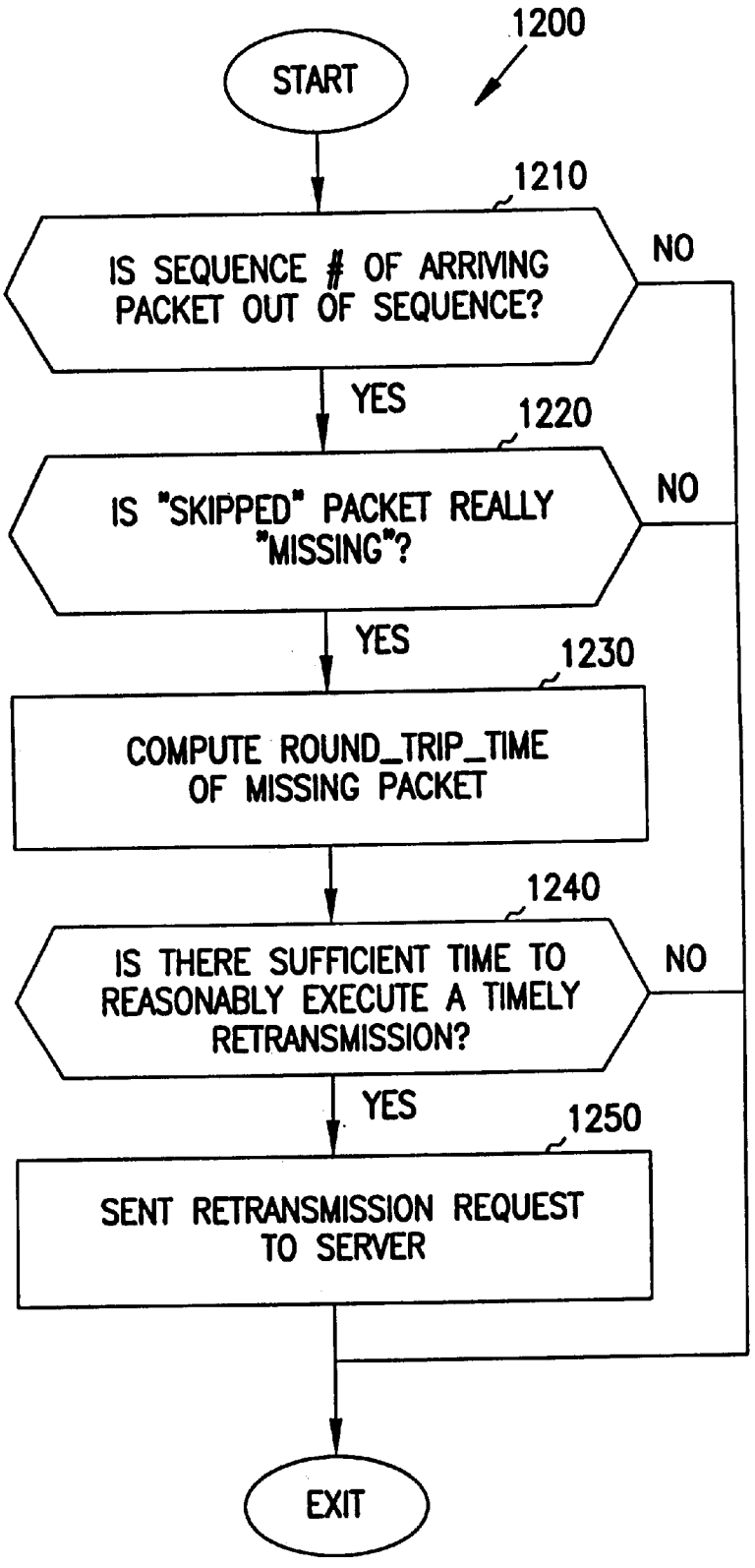

1200
START
1210
IS SEQUENCE # OF ARRIVING PACKET OUT OF SEQUENCE?
NO
YES
1220
IS "SKIPPED" PACKET REALLY "MISSING"?
NO
YES
1230
COMPUTE ROUND_TRIP_TIME OF MISSING PACKET
1240
IS THERE SUFFICIENT TIME TO REASONABLY EXECUTE A TIMELY RETRANSMISSION?
NO
YES
1250
SENT RETRANSMISSION REQUEST TO SERVER
EXIT

FIG. 12

DYNAMIC BANDWIDTH SELECTION FOR EFFICIENT TRANSMISSION OF MULTIMEDIA STREAMS IN A COMPUTER NETWORK

RELATED APPLICATIONS

This application is related to U.S. application Ser. No. 08/818,805, filed on Mar. 14, 1997, entitled "Method and Apparatus for Implementing Motion Detection in Video Compression," U.S. application Ser. No. 08/819,507, filed Mar. 14, 1997, entitled "Digital Video Signal Encoder and Encoding Method," U.S. application Ser. No. 08/818,804, filed on Mar. 14, 1997, entitled "Production of a Video Stream with Synchronized Annotations over a Computer Network," U.S. application Ser. No. 08/819,586, filed on Mar. 14, 1997, entitled "Method and apparatus for Implementing Control Functions in a Streamed Video Display System," U.S. application Ser. No. 08/818,769, filed on Mar. 14, 1997, entitled "Method and apparatus for Automatically Detecting Protocols in a Computer Network," U.S. application Ser. No. 08/818,127, filed on Mar. 14, 1997, entitled "Dynamic Bandwidth Selection for Efficient Transmission of Multimedia Streams in a Computer Network," U.S. application Ser. No. 08/819,585, filed on Mar. 14, 1997, entitled "Streaming and Display of a Video Stream with Synchronized Annotations over a Computer Network," U.S. application Ser. No. 08/818,664, filed on Mar. 14, 1997, entitled "Selective Retransmission for Efficient and Reliable Streaming of Multimedia Packets in a Computer Network," U.S application Ser. No. 08/819,579, filed Mar. 14, 1997, entitled "Method and apparatus for Table-Based Compression with Embedded Coding," U.S. application Ser. No. 08/819,587, filed Mar. 14, 1997, entitled "Method and apparatus for Implementing Motion Estimation in Video Compression," U.S. application Ser. No. 08/818,826, filed on Mar. 14, 1997, entitled "Digital Video Signal Encoder and Encoding Method," all filed concurrently herewith, U.S. application Ser. No. 08/822,156, filed on Mar. 17, 1997, entitled "Method and apparatus for Communication Media Commands and Data Using the HTTP Protocol," U.S. provisional application Serial No. 60/036, 662, filed on Jan. 30, 1997, entitled "Methods and apparatus for Autodetecting Protocols in a Computer Network," U.S. application Ser. No. 08/625,650, filed on Mar. 29, 1996, entitled "Table-Based Low-Level Image Classification System," U.S. application Ser. No. 08/714,447, filed on Sep. 16, 1996, entitled "Multimedia Compression System with Additive Temporal Layers," and is a continuation-in-part of U.S. application Ser. No. 08/623,299, filed on Mar. 28, 1996, entitled "Table-Based Compression with Embedded Coding," which are all incorporated by reference in their entirety for all purposes.

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to multimedia communications. More particularly, the present invention relates to the efficient and reliable delivery of multimedia streams over a diverse computer network with dynamically variable bandwidth capacity.

2. Description of the Related Art

With the proliferation of connections to the internet by a rapidly growing number of users, the viability of the internet as a widely accepted medium of communication has increased correspondingly. Bandwidth requirements can vary significantly depending on the type of multimedia data being delivered. For example, a low resolution, low frame rate video telephone call may require only an ISDN connection, while a high resolution video broadcast of a live event to a large group of viewers may require the bandwidth of a T1 connection to the server. Hence, the ability to efficiently deliver multimedia data over a diverse computer network such as the internet is severely limited by the reliability and bandwidth capacity of the network connection.

The first problem is the average transmission capacity. In an ideal packet-based delivery system with an input buffer at a client computer, data packets arrive at the client computer in the same order and at the same interval the packets were sent by the server. In an ideal example, at time t, the client computer receives data packet #1 with a time stamp of 0.0 second. Subsequently, at time t+1 second, data packet #2 with a time stamp of 1.0 second arrives, followed by data packet #3 with a time stamp of 2.0 seconds at time t+2 seconds. As a result, packets arrive at and are consumed by the client computer at the same rate as they were sent.

However in a more realistic example, the network connection may be unable to keep up with the demands of the server/client stream traffic, i.e., the average bandwidth capacity of the network connection may be insufficient. Consequently, data packets will arrive at the client computer later and later in time, causing the input buffer to empty at a faster rate than it can be replenished, and eventually depleting the input buffer. For example, data packet #2 with the time stamp of 1.0 seconds may arrive at time t+1.2 seconds, followed by data packet #3 with the time stamp of 2.0 seconds arriving at time t+2.4 seconds. In other words, the average bandwidth capacity of the network connection is insufficient to support the transmission rate selected by the server/client. This is a first order bandwidth capacity problem.

The second problem is the rate of change of bandwidth capacity over time of the network connection. Since overall traffic within the internet is not constant, and since the internet is packet-switched, the bandwidth capacity provided by the internet for the network connection can vary dynamically over time. Accordingly, if an application is too aggressive in demanding bandwidth, during peak demand periods, the internet may be unable to cope with the peak demand, causing packets to be discarded/lost and requiring retransmission, which further degrades the overall performance of the network connection. This is a second order network bandwidth problem, i.e., changes in the bandwidth capacity over time.

In a real time application, e.g., a video on demand (VOD) application, the discarded/lost packets result in jitter. Jitter is defined as the second order timing difference in the packet arrival times. In the ideal example, where packet #2 and packet #3 arrive at t+1.0 second and t+2.0 seconds, respectively, jitter is zero., because the inter-arrival times, i.e., differences in arrival times, are identical.

However, in a more realistic example, packets #2 and #3 may arrive at t+0.9 second and t+2.1 seconds, with inter-arrival times of 0.9 second and 1.2 seconds, respectively. Although the input buffer of the client computer provides partial relief by buffering the incoming packets and releasing them to applications on the client computer at a less jittery rate, unfortunately, in the real time application, the length of the input buffer has to be kept to a minimum, thereby severely limiting the relief attainable.

In view of the foregoing, there are desired improved techniques for reliable and efficient transmission of multimedia streams to client(s) which efficiently utilize the network resources available over a period of time.

SUMMARY OF THE INVENTION

The present invention provides efficient transmission of multimedia streams from a server to a client computer over a diverse computer network including local area networks (LANs) and wide area networks (WANs) such as the internet. Examples of multimedia streams provided to the client computer include a compressed video stream, a compressed audio stream, and an annotation stream with pointers to textual/graphical data in the form of HTML pages.

In one embodiment, the client computer includes a playout buffer, and the transmission rate is dynamically matched to the available bandwidth capacity of the network connection between the server and the client computer.

If a playtime of the playout buffer, which is one measure of the number of data packets currently in the playout buffer, drops below a dynamically computed Decrease_Bandwidth (DEC_BW) threshold, then the transmission rate is decreased by sending a DEC_BW message to the server.

Conversely, if the number of packets remaining in the playout buffer rises above a dynamically computed Upper Increase_Bandwidth (INC_BW) threshold and does not drop below a Lower INC_BW threshold for at least an INC_BW wait period, then the transmission rate is incremented.

In this embodiment, the transmission rate is selected from among a predetermined set of discrete bandwidth values. However the invention is also applicable to a system in which the transmission rate is selected from within a continuous range of bandwidth values.

In another embodiment, in addition to responding to variations in network connection capacity, the client computer also determines an average client computational capacity. Accordingly, if the average client computational capacity is less than the network capacity, the lower of the two capacities is the determining one, thereby avoiding a playout buffer overrun.

These and other advantages of the present invention will become apparent upon reading the following detailed descriptions and studying the various figures of the drawings.

BRIEF DESCRIPTION OF THE DRAWING

FIG. **1** is a block diagram of an exemplary computer system for practicing the various aspects of the present invention.

FIG. **2** is a block diagram showing an exemplary hardware environment for practicing the reliable and efficient video-on-demand (VOD) system of the present invention.

FIG. **3** is a block diagram showing a producer which includes a capture module and an author module for capturing video streams and for generating annotation streams, respectively.

FIG. **4** is a flowchart including steps **410**, **420**, **430**, **440** and **450** which illustrate the Adjust_Bandwidth procedure of one embodiment of the present invention.

FIG. **5**A, **5**B, **5**C, **5**D and **5**E, are detailed flowcharts illustrating steps **410**, **420**, **430**, **440** and **450**, respectively, of FIG. **4**.

FIGS. **6**A and **6**B are two halves of a flowchart illustrating the dynamic determination of the Upper INC_BW threshold and the DEC_BW threshold.

FIG. **7**A is a flowchart illustrating the computation of variables Playtime and Delta_Playtime of the playout buffer.

FIG. **7**B illustrates the determination of the Duetime of a data packet.

FIG. **8** is a flowchart showing the determination of the Round_Trip_Time_Bit.

FIG. **9** is a flowchart showing the determination of the Lossrate_Bit.

FIG. **10** illustrates a periodic update of Lossrate.

FIG. **11** is a flowchart showing a dynamic bandwidth selection which optimizes the computational capacity of the client computer and which is also sustainable by the network connection.

FIG. **12** is a flowchart illustrating the selective retransmision of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention will now be described in detail with reference to a few preferred embodiments thereof as illustrated in the accompanying drawings. In the following description, numerous specific details are set forth in order to provide a thorough understanding of the present invention. It will be apparent, however, to one skilled in the art, that the present invention may be practiced without some or all of these specific details. In other instances, well known process steps have not been described in detail in order to not unnecessarily obscure the present invention.

FIG. **1** is a block diagram of an exemplary computer system **100** for practicing the various aspects of the present invention. Computer system **100** includes a display screen (or monitor) **104**, a printer **106**, a floppy disk drive **108**, a hard disk drive **110**, a network interface **112**, and a keyboard **114**. Computer system **100** includes a microprocessor **116**, a memory bus **118**, random access memory (RAM) **120**, read only memory (ROM) **122**, a peripheral bus **124**, and a keyboard controller **126**. Computer system **100** can be a personal computer (such as an Apple computer, e.g., an Apple Macintosh, an IBM personal computer, or one of the compatibles thereof), a workstation computer (such as a Sun Microsystems or Hewlett-Packard workstation), or some other type of computer.

Microprocessor **116** is a general purpose digital processor which controls the operation of computer system **100**. Microprocessor **116** can be a single-chip processor or can be implemented with multiple components. Using instructions retrieved from memory, microprocessor **116** controls the reception and manipulation of input data and the output and display of data on output devices.

Memory bus **118** is used by microprocessor **116** to access RAM **120** and ROM **122**. RAM **120** is used by microprocessor **116** as a general storage area and as scratch-pad memory, and can also be used to store input data and processed data. ROM **122** can be used to store instructions or program code followed by microprocessor **116** as well as other data.

Peripheral bus **124** is used to access the input, output, and storage devices used by computer system **100**. In the described embodiment(s), these devices include display screen **104**, printer device **106**, floppy disk drive **108**, hard disk drive **110**, and network interface **112**. Keyboard controller **126** is used to receive input from keyboard **114** and send decoded symbols for each pressed key to microprocessor **116** over bus **128**.

Display screen **104** is an output device that displays images of data provided by microprocessor **116** via peripheral bus **124** or provided by other components in computer

system **100**. Printer device **106** when operating as a printer provides an image on a sheet of paper or a similar surface. Other output devices such as a plotter, typesetter, etc. can be used in place of, or in addition to, printer device **106**.

Floppy disk drive **108** and hard disk drive **110** can be used to store various types of data. Floppy disk drive **108** facilitates transporting such data to other computer systems, and hard disk drive **110** permits fast access to large amounts of stored data.

Microprocessor **116** together with an operating system operate to execute computer code and produce and use data. The computer code and data may reside on RAM **120**, ROM **122**, or hard disk drive **120**. The computer code and data could also reside on a removable program medium and loaded or installed onto computer system **100** when needed. Removable program mediums include, for example, CD-ROM, PC-CARD, floppy disk and magnetic tape.

Network interface circuit **112** is used to send and receive data over a network connected to other computer systems. An interface card or similar device and appropriate software implemented by microprocessor **116** can be used to connect computer system **100** to an existing network and transfer data according to standard protocols.

Keyboard **114** is used by a user to input commands and other instructions to computer system **100**. Other types of user input devices can also be used in conjunction with the present invention. For example, pointing devices such as a computer mouse, a track ball, a stylus, or a tablet can be used to manipulate a pointer on a screen of a general-purpose computer.

The present invention can also be embodied as computer readable code on a computer readable medium. The computer readable medium is any data storage device that can store data which can be thereafter be read by a computer system. Examples of the computer readable medium include read-only memory, random-access memory, magnetic data storage devices such as diskettes, and optical data storage devices such as CD-ROMs. The computer readable medium can also be distributed over a network coupled computer systems so that the computer readable code is stored and executed in a distributed fashion.

FIG. **2** is a block diagram showing an exemplary hardware environment for practicing the reliable and efficient video-on-demand (VOD) system of the present invention. The VOD system includes a production station **210**, a stream server **220**, at least one web server **230** and at least one client computer **240**, each of which can be implemented using computer system **100** described above. Stream server **220** and web server **230** are coupled to client computer **240** via a computer network **290**, e.g., the internet. Note that the disclosed hardware environment is exemplary. For example, production station **210** and stream server **220** can be implemented using two separate computer systems or using one computer system. In addition, if production station **210** and stream server **220** are implemented on separate computer systems as shown in FIG. **2**, an optional direct connection (not shown) between production station **210** and stream server **220** can provide faster uploads of compressed video and annotation streams. In the following description, an audio stream optionally accompanies each video stream.

A producer **215**, installed in production station **210**, is a user-friendly tool for use by a designer **219** to create a synchronization script which includes annotation stream(s). The annotation stream(s) define the content(s) of a LiveScreen display **245** to be displayed on client computer **240** for a viewer **249**. LiveScreen **245** display provides a graphical user interface (GUI) with multiple windows for synchronously displaying a video stream from stream server **220** and at least one displayable event stream. Examples of displayable events include textual/graphical information such as HTML-scripted web page(s) from web server **230**.

Referring to FIG. **3**, producer **215** includes a capture module **317** and an author module **318**. Production station **210** includes 16 MB of RAM and a 1 GB hard disk drive for capturing and storing an uncompressed or precompressed video stream. Sources for generating video streams include a video camera **312**, a video cassette recorder (VCR) (not shown) or a previously digitized video file **314**, e.g., a Video for Windows (.avi) file. For ease of installation and use by designer **219**, producer **215** is implemented in a host environment which includes a window-based operating system such as Microsoft Windows 95 and a web browser such as Netscape's Navigator 2.x.

Client computer **240** in FIG. **3** includes a web browser **350** and a browser plug-in module **352** for interfacing web browser **350** with a main client module **360**. Client module **360** includes an event registry **362**, video/audio decoder(s) **364**, video/audio renderer(s) **365**, playout buffer(s) **366**, and one or more dynamically loadable event applet(s), e.g., flipper applet **367**, ticker applet **368** and VCR applet **369**. In this embodiment, event registry **362** also functions as an annotation interpreter **363**.

Co-pending applications **702**, **712** and **718** provides a detailed description of the decompression and rendering of the video/audio streams at client computer **240** once the streamed packets have arrived from stream server **220**.

The present invention is directed at the efficient and reliable streaming of data packets from stream server **220** to client computer **240**, accomplished by optimally utilizing the bandwidth of the connection provided by computer network **290** while minimizing the loss of packets. In one embodiment, the transmission rate of the data stream is dynamically adjusted in response to changes in the bandwidth made available by computer network **290** for the network connection between server **220** and client computer **240**. Accordingly, server **220**, in response to feedback from client computer **240**, dynamically selects transmission rates in order to better match the varying bandwidth capacity of the network connection. For example, server **220** streams video packets at 1 frames/second (fps), 5 fps, 10 fps, and 15 fps for bandwidths of 4 kbits/second (kbps), 14 kbps, 18 kbps, and 44 kbps.

In this embodiment, client module **360** includes playout buffer **366** which stores several seconds, e.g., 5 seconds, worth of data packets from the data stream. The buffer **366** enables data packets to independently traverse computer network **290**, arrive at client computer **240** in a different order than they were originally transmitted, and be rearranged back to their original sequential order prior to processing by decoder **364** and then renderer **365**. Playout buffer **366** also enables retransmitted (lost) packets to be inserted in their originally sequential order prior to processing by decoder **364**. A suitable reordering algorithm, such as a "map" object from the Standard Template Library (STL) Toolkit can be used to reorder and/or insert packets in buffer **366**. Accordingly, suitable data structures for playout buffer **366** include STL maps and linked lists.

In accordance with one aspect of the present invention, client computer **240** dynamically adjust the transmission rate of the data stream to optimize usage of the bandwidth capacity of the network connection. Note that in the following examples, within the context of server **220** and client

computer **240**, the term "bandwidth" is synonymous to the "transmission rate". FIG. **4** is a flowchart including steps **410**, **420**, **430**, **440** and **450** which illustrate the Adjust_Bandwidth procedure of one embodiment of the present invention.

In this example, the performance bottleneck is the bandwidth capacity of the network connection, and a transmission rate, sustainable by the network connection, is dynamically selected from a plurality of discrete bit rates, e.g., 14 kbps, 18 kbps, . . . These bit rates are exemplary and other discrete bit rate values are possible. In addition, the present method for dynamically selecting a suitable bandwidth among a plurality of discrete bit rate values can also be adapted for dynamically selecting a suitable bandwidth within a continuous range of bit rate values.

FIGS. **5**A, **5**B, **5**C, **5**D and **5**E, are detailed flowcharts illustrating steps **410**, **420**, **430**, **440** and **450**, respectively, of FIG. **4**. In step **410**, the performance variables are computed. Next, in step **420**, the computed performance variables are used to determine if it is desirable to decrease the bandwidth, and if so, then in step **430**, the bandwidth is decreased. If a bandwidth decrease is not desirable, then in step **440**, the performance variables are used to determine if it is desirable to increase the bandwidth. If a bandwidth increase is desirable, then in step **450**, the bandwidth is increased.

Referring now to FIG. **5**A (step **410**), performance variables are computed. In step **512**, an Upper Increase_Bandwidth (INC_BW) threshold and a Decrease_Bandwidth (DEC_BW) threshold are computed. Next, variables Playtime and Delta_Playtime are computed (step **513**). In steps **514**, **516**, client computer **240** determines if Round_Trip_Time_Bit and Lossrate_Bit are high. In step **518**, the Lossrate is updated periodically.

FIGS. **6**A and **6**B are two halves of a flowchart illustrating the dynamic determination of the Upper INC_BW threshold and the DEC_BW threshold, step **512** in greater detail. In step **612**, the difference (D1) between the Current_Time and the previous time the dynamic bandwidth selection method was invoked is computed. In step **614**, the difference (D2) between the timestamp of the last data packet currently in playout buffer **366** and the timestamp of the last data packet in playout buffer **366** during the previous invocation of the Adjust_Bandwidth procedure, is computed. In step **616**, the difference (D3) between the number of bytes received by the previous invocation and the number of bytes currently received (by the current invocation) is computed.

If D1 is greater than a constant C9, e.g., 2.5 seconds, and D2 is greater than a constant C10, e.g., 1.5 seconds, (steps **622** & **624**), then client computer **240** computes an average of the last C11 samples, e.g., ten samples, of the quotient (Q1) from a division of D3 by D2 (step **632**). If D1 is greater than zero (step **634**), then an average of the last C11 samples of the quotient (Q2) from a division of D3 by D1 is computed (step **638**).

If C11 is greater than C12, e.g., if the number of samples is greater than 3, then the DEC_BW threshold and the Upper INC_BW threshold are dynamically adjusted using the following equations 650, 660, 670 & 690:

```
DEC_BW threshold (eqn 650) :=
    ((((Ideal_Playout_Buffer_size) +
    (Codec_Specific_Constant)) *((Average of Q1) - (Average of Q2)))
    /(Average of Q2)) + ((Ideal_Playout_Buffer_Size) * C13)
DEC_BW threshold (eqn 690) :=
        Max (DEC_BW threshold, DEC_BW threshold * C17)
Upper INC_BW threshold (eqn 660) := (Ideal_Playout_Buffer_Size) -
        ((Average_Packet_Size) / (Average of Q2))
Upper INC_BW threshold (eqn 670) :=
        Max (Min (Upper INC_BW threshold,
            (Ideal_Playout_Buffer_Size) * C14),
                (Ideal_Playout_Buffer_Size) * C15)
```

Wherein, C13 is 0.25, C14 is 0.95, C15 is 0.60, and C17 is 0.20. The Codec_Specific_Constant is dependent on the specific codec, e.g., for H263 the constant is 6400 milliseconds (ms).

Else if C11 is less than C12, then the DEC_BW threshold and the Upper INC_BW threshold are dynamically adjusted using the following equations 680, 690:

```
Upper INC_BW threshold (eqn 680) :=
        (Ideal_Playout_Buffer_Size) * C16
DEC_BW threshold (eqn 690) :=
        Max (DEC_BW threshold, DEC_BW threshold * C17)
```

Wherein C14 is 0.95, C15 is 0.60, and C16 is 0.60.

FIG. **7**A is a flowchart illustrating the computation of variables Playtime and Delta_Playtime, step **513**, in greater detail. In step **710**, Playtime is set to the Duetime of the last packet in playout buffer **366**. The computation of the Duetime is described in greater detail in step **740** below. Client computer **240** determines the change in the Playout_Buffer_Size (step **720**). The Delta_Playtime is set to the difference between the current Playtime and the Playtime at the previous invocation of the Adjust_Bandwidth procedure (step **730**). Variables Playtime and Delta Playtime provide exemplary absolute and relative measures, respectively, of the Playout_Buffer_Size, the number of data packet(s) in playout buffer **366**.

FIG. **7**B illustrate the determination of the Duetime of a data packet (step **710**). First, the Base_TS is set to the timestamp of the first packet received by client computer **240** (step **712**). The Base_Time is set to the time when the first packet was received (step **716**). The TS is set the timestamp of the data packet of interest (step **746**). The Duetime of the packet of interest is computed using the following equation 718:

Duetime:=(Ideal_Playout_Buffer_Size)+(TS−Base_TS)−(Current_Time−Base_Time)

As shown in FIG. **8**, in step **514**, client computer **240** determines if Round_Trip_Time_Bit should or should not be set to High. The boolean equation 810 used for the determination is:

```
(Round_Trip_Time>C18) &
(Round_Trip_Time has increased over the last C19 samples) &
  (New sampling of Round_Trip_Time occurred since the previous
                Reduce_Bandwidth message was sent to the server
because the     Round_Trip_Time_Bit was set to High)
  Wherein C18 is 4 seconds, and C19 is 3 samples.
```

In step **516** of FIG. **9**, a determination of whether Lossrate_Bit should be set to High. The boolean equation 910 used is for the determination:

(Number of samples of Lossrate>C20) & (Lossrate>C21) & (a new sample of Lossrate was taken since the last

Reduce_Bandwidth message was sent to the server because the Lossrate_Bit was High)

FIG. **10** shows step **518** which updates Lossrate periodically. In steps **1010** and **1020**, Expected_Last is set to the maximum sequence number among the packets received when Lossrate was last computed, and Expected is set to the maximum sequence number among the packets currently received. Received_Last is set to the total number of packets received when Lossrate was last computed (step **1030**), and Received is set to the total number of packets currently received (step **1040**).

The Lossrate is then computed using the following equation 1050:

Lossrate:=(((Expected_Expected_Last)_Received_Received_Last))*100)/Expected_Expected_Last)

Referring back to FIG. **5**B, in step **420**, client computer **240** uses the performance variables to determine if the bandwidth should be decreased. In this implementation, a conservative approach is taken, i.e., the bandwidth is decreased whenever a bandwidth reduction appears to be required. Such a conservative approach reduces the probability of an overrun of playout buffer **366** and the consequential loss of packets.

Using the boolean equation 522:

((Delta_Playtime<C1)&(Playtime<DEC_BW threshold)) OR (Round_Trip_Time_Bit=High) OR (Lossrate_Bit=(High)

Wherein C1=100 ms

If equation 522 is True, then playout buffer **366** is permitted to stablize prior to sending any successive Decrease_Bandwidth messages. As discussed above, Playtime and Delta_Playtime provide measures of the number of packet(s) in playout buffer **366**.

If Playtime did not increase past the Upper INC_BW threshold since the previous Reduce_Bandwidth message was sent (step **524**), then client computer **240** permits playout buffer **366** to stabilize at the current bandwidth.

If client computer **240** has not previously sent a Decrease_Bandwidth message or client computer **240** has sent an Increase_Bandwidth message since the last Decrease_Bandwidth message was sent (step **526**), then step **430** is invoked (**420***y*). Conversely, if the difference between the Current_Time and the time the last Decrease_Bandwidth message was sent is greater than the sum of the Ideal_Buffer_Size and the average Round_Trip_Time to stream server **220** (step **528**), then step **430** is invoked (**420***y*).

As shown in FIG. **5**C, in step **430**, client computer **240** has determined that a network bandwidth decrease is desired (**420***y*). A Time_Before_Increase variable is maintained for each discrete bandwidth point (discrete bandwidth value), and are initialized to a suitable value, C5, e.g., 10 seconds (step **531**). For each bandwidth point, the Time_Before_Increase value determines the time period for which Playout_Buffer_Size should stay above a Lower INC_BW threshold, e.g., 75% of the Ideal_Playout_Buffer_Size, before an Increase_Bandwidth message is sent. In other words, for each bandwidth point, the variable Time_Before_Increase determines the minimum waiting period prior to the sending of an Increase_Bandwidth message to server **220**.

Client computer **240** determines the time, T1, when an Increase_Bandwidth message from a particular bandwidth was sent (step **532**). If the present bandwidth reduction reached the particular bandwidth (step **533**), then client computer **240** computes the difference between the time of such a reduction and T1 (step **534**), else client computer **240** sends a Decrease_Bandwidth message to stream server **220** (step **537**). The Adjust_Bandwidth procedure is now completed for the current invocation.

In step **534**, if the computed difference between the time of the reduction and T1 is less than C2, e.g., 80 seconds, then for the particular bandwidth, the Time_Before_Increase is set to the maximum of (C3, C4 * Time_Before_Increase) (step **535**). Conversely, if the difference is greater than C2, then the Time_Before_Increase is reset to C5, e.g., 10 seconds (step **536**). In this example, C3=180 and C4=1.75.

After step **535** or **536**, client computer **240** sends a Decrease_Bandwidth message to stream server **220** (step **537**). If the underlying transmission protocol between client computer **240** and server **220** is HTTP, then commands, such as the Decrease_Bandwidth message can be sent from client computer **240** to stream server **220** within a HTTP "post" packet. The Adjust_Bandwidth procedure is now completed for the current invocation.

Referring now to FIG. **5**D, if a bandwidth decrease is not desirable (**420***n*), then in step **440**, the performance variables are used to determine if it is desirable to increase the bandwidth. In this conservative approach, if the Playout_Buffer_Size exceeds the Upper INC_BW threshold and continues to stay above the Lower INC_BW threshold for the INC_BW wait period, then the bandwidth is increased. In other words, the bandwidth is increased only when there is a fairly high probability that the next higher bandwidth will be sustainable by computer network **290**. Hence, the Lower_INC_BW threshold requirement reduces the probability of the selected bandwidth oscillating rapidly between two bandwidth points and possibly causing jitter.

Accordingly, in step **541**, if Playtime is greater than the Upper INC_BW threshold, then the Time_Buffer_Full is set to the time when the Playout Buffer_Size first increased past the Upper INC_BW threshold (step **542**). In step **543**, whenever the Playout_Buffer_Size drops below the Lower INC_BW threshold, e.g., 75% of the Ideal_Playout_Buffer_Size, the Time_Buffer_Full is reset to zero. Next, client computer **240** determines if the following Boolean equation 544 is True:

(# of Decrease_Bandwidth message(s) sent to the server is greater than # of Increase_Bandwidth message(s) sent to the server) &

(Difference between the Current_Time & the last time the bandwidth was switched (to the current bandwidth) is greater than the Time_Before_Increase) &

(The average Lossrate is less than C6)

Wherein C6=10

If equation 544 is true, then client computer **240** determines if the previous reduction of bandwidth was because the Lossrate_Bit was High and the average Lossrate is less than C7, e.g., 5 (step **545**). If step **545** is true, client computer **240** proceeds with an increase of the bandwidth (**440***y*).

Conversely, if step **545** was not True, then client computer **240** determines if the following boolean equation 546 is True:

(Previous bandwidth switch was not due to a High Lossrate_Bit) &

(Average Lossrate is less than C8) &

((Difference between the Current_Time and the Time_Buffer_Full) is greater than (the Time_Before_Increase))

wherein C8=10

If equation 546 is True, then client computer **240** proceeds with an increase of the bandwidth (**440***y*). Otherwise, the

Adjust_Bandwidth procedure is now completed for the current invocation.

In FIG. **5**E, if a bandwidth increase is desirable (**440***y*), then in step **450**, the bandwidth is increased. Client computer **240** sends an Increase_Bandwidth message to stream server **220**. The Adjust_Bandwidth procedure is now completed for the current invocation.

In accordance with another aspect of the present invention, as shown in FIG. **11**, client computer **240** dynamically selects a suitable bandwidth which optimizes the computational capacity of client computer **240** and which is also sustainable by the network connection. In this example, the bottleneck is the client computer's computational bandwidth.

First, client computer computes the performance variables (step **410** of FIG. **4**), and also computes an average client packet computational rate which is the rate client computer **240** is able to decompress and render all the incoming data packets without loss (step **1110**). Next, client computer **240** determines if the average client packet computational rate is higher than, equal to or lower than the selected bandwidth, i.e., selected transmission rate (step **1120**). In other words, on the average, are the data packets arriving at client computer **240** at a faster, equal or slower rate than client computer **240** is able to decompress and render the data packets.

If the selected bandwidth is higher than the average client computation rate, buffer **366** will eventually overflow and data packets will have to be discarded. According, a lower bandwidth, less than or equal to the average computation rate of client computer **240**, is selected (step **1130**). Such a bandwidth decrease can be implemented using the method described above and illustrated by steps **420**, **430** of the flowchart of FIG. **4**.

Conversely, if the average client packet computational rate is higher than the selected bandwidth, then a higher bandwidth may be selected, subject to the bandwidth capacity of the network connection (step **1140**). Such a network bandwidth increase can be implemented using the method described above and illustrated by steps **440**, **450** of the flowchart of FIG. **4**.

In accordance with yet another aspect of the invention, as shown in FIG. **12**, client computer **240** selectively requests retransmission of "missing" data packets for just-in-time (JIT) reliability. As data packets arrive at client computer **240**, their sequence numbers are checked (step **1210**). If a data packet arrives out of sequence, e.g., data packet # n+2 arrives after data packet # n, client computer **240** checks playout buffer **366** to see if the "skipped" packet, e.g., packet # n+1, is indeed "missing", or if the skipped packet has arrived previously and is already stored in playout buffer **366** (step **1220**).

If the skipped data packet # n+1 is not found in playout buffer **366**, i.e., packet # n+1 is missing, client computer **240** computes a Round_Trip_Time for the missing data packet # n+1. The Round_Trip_Time is an estimate of the time beginning from the time a retransmission request is sent to stream server **220** till the time a copy of the missing data packet is received at client computer **240** in response to the retransmission request (step **1230**).

If there is sufficient time to reasonably execute a timely retransmission, e.g., the difference between the timestamp of the missing data packet and the timestamp of the currently displayed data packet is greater than the Round_Trip_Time (step **1240**), then client computer **240** sends a request to server **220** for a retransmission of the missing data packet (**1250**).). As discussed above, if the underlying transmission protocol between client computer **240** and server **220** is HTTP, then commands, such as the retransmission request can be sent from client computer **240** to stream server **220** within a HTTP "posf" packet.

Conversely, if there is insufficient time remaining to reasonably expect a timely retransmission, then the data packet is presumed "unrecoverable". By selectively requesting retransmission, data packets which do not have a reasonable chance for a successful retransmission are discarded, thereby reducing network traffic due to late retransmissions and further improving network efficiency.

The above described selective retransmission algorithm, an application-level framing (ALF) based algorithm, is advantageous over a conventional automatic retransmission algorithm based on a full-blown multi-layer protocol model, e.g., the OSI 7-layer networking model, with accompanying strictly layered functional divisions and related interfaces. This is because vertical control across a structured multi-layer protocol is not easy to implement nor efficient. In contrast, a simple protocol, e.g., RTP over UDP without high level integrated packet reliability, is easier to implement efficiently than for example TCP or HTTP over TCP.

The present invention may also be practiced with the prioritization of retransmission based on data types of the data packets. For example, since parent I frames are needed to render dependent child P frames, data packets which include I frame(s) should assigned higher priority for transmission and/or retransmission over data packets which include only P frame(s).

Priority can also be determined via a scalable layered protocol based on parameters such as resolution or frame rate. For example, the data packets for rendering the base (lowest) resolution is given the highest priority while additive data packets for improving the base resolution are given lower priority.

Other modifications to the above described algorithm is also possible. For example, instead of the less flexible rule of step **1140**, the present invention may also incorporate selective late retransmission. Hence, even when there is insufficient time remaining for a timely retransmission, instead of dropping the missing data packet, client computer **240** may decide to temporarily halt the video/audio stream for a brief interval to wait for retransmission of the relatively important missing data packet. For example, if the audio stream is compressed and packaged into 1.6 to 2 seconds sized packet, it is preferable to pause the video/audio streams for 0.2 seconds than to lose over a second of audio information.

Similarly, since it is visually acceptable to momentarily freeze a ticker tape display or slightly delay a HTML page change, annotation stream packets are also suitable for selective late retransmissions. For example, a late retransmission of a missing annotation stream packet, which includes important HTML flip(s), may be preferred to dropping the important annotation stream packet.

Data-type adaptability can also be incorporated into the selective retransmission protocol of the present invention. For example, if there is a missing P frame which is sequentially located just before an I frame, it may be expedient to drop the missing P frame and skip to the next I frame in buffer **366**.

While this invention has been described in terms of several preferred embodiments, other alterations, permutations, and equivalents also fall within the scope of this invention. For example, one conservative approach is to start at a very low bandwidth and slowly increase the bandwidth. Another approach is to be optimistic and start at

a high bandwidth and then rapidly decrease the bandwidth to match the network capability. Hence, there are many alternative ways of implementing the methods and apparatuses of the present invention. It is therefore intended that the following appended claims be interpreted as including all such alterations, permutations, and equivalents as fall within the true spirit and scope of the present invention.

What is claimed is:

1. In a client computer, a method of dynamically adjusting a data transmission rate of a data stream from a server to the client computer, the client computer coupled to the server via a computer network, the computer network providing a network connection with a variable bandwidth for transmitting the data stream, the data stream including a plurality of data packets, said client computer including a playout buffer for buffering the incoming data packets, the method comprising:

initializing the data transmission rate for said data stream;

dynamically computing a decrement bandwidth (DEC_BW) threshold for the playout buffer; and

decrementing the transmission rate upon determining that a difference between a first time of a first data packet and a last time of a last data packet in said playout buffer drops below said dynamically computed DEC_BW threshold.

2. The method of claim **1** wherein said current number of packets remaining is represented by a current playtime which is set to a duetime of a last data packet in said playout buffer.

3. The method of claim **1** further comprising:

determining a lossrate of the data packets; and

decrementing the transmission rate upon determining the lossrate exceeds a lossrate threshold.

4. The method of claim **1** wherein said data stream is a video data stream.

5. The method of claim **1** further comprising:

determining an average client computational rate of said client computer; and

decrementing the transmission rate upon determining the average client computational rate is less than the transmission rate.

6. The method of claim **1**, further comprising decrementing the transmission rate upon determining a difference between the current number of data packets and a previous number of data packets remaining in said playout buffer is less than a first constant, then decrementing the transmission rate.

7. In a client computer, a method of dynamically adjusting a transmission rate of a data stream from a server to the client computer, the client computer coupled to the server via a computer network, the computer network providing a network connection with a variable bandwidth for transmitting the data stream, the data stream including a plurality of data packets, said client computer including a playout buffer for buffering the incoming data packets, the method comprising:

initializing the transmission rate for said data stream;

initializing a Time_Before_Increase;

dynamically computing an upper increment bandwidth (INC_BW) threshold and a lower INC_BW threshold for said playout buffer; and

incrementing the transmission rate upon determining a current number of data packets remaining in the playout buffer rises above the upper INC_BW threshold and does not drop below the lower INC_BW threshold for at least said Time_Before_Increase.

8. The method of claim **7** wherein a Time_Buffer_Full indicates how long the current number of data packets remaining in the playout buffer has risen above the upper INC_BW threshold, and wherein the method further comprising:

resetting the Time_Buffer_Full upon determining the current number of data packets remaining in the playout buffer rises above the upper INC_BW threshold and then drops below the lower INC_BW threshold within said Time_Before_Increase.

9. The method of claim **7** further comprising increasing said Time_Before_Increase upon determining the transmission rate is reduced shortly after it was previously increased.

10. The method of claim **7** wherein said lower INC_BW threshold is initialized to a percentage of an ideal playout buffer size.

11. The method of claim **7** further comprising the steps of:

determining an average client computational rate of said client computer; and

if the average client computational rate is greater than the transmission rate and if the current number of data packets remaining in the playout buffer rises above the upper INC_BW threshold and does not drop below the lower INC_BW threshold for at least said INC_BW wait period, then incrementing the transmission rate.

12. The method of claim **7** wherein said data stream is a video data stream.

13. The method of claim **7** wherein said current number of packets remaining is represented by a current playtime which is set to a duetime of a last data packet in said playout buffer.

14. The method of claim **7** further comprising:

dynamically computing a decrement bandwidth (DEC_BW) threshold for the playout buffer; and

decrementing the transmission rate upon determining the current number of data packets remaining in said playout buffer drops below said dynamically computed DEC_BW threshold, and upon determining a difference between the current number of data packets and a previous number of data packets remaining in said playout buffer is less than a first constant.

15. A client computer useful in association with a server, the client computer coupled to the server via a computer network, the computer network providing a network connection with a variable bandwidth for transmitting a data stream from the server to the client computer, the data stream including a plurality of data packets, the client computer comprising:

a playout buffer for buffering the incoming data packets, wherein a transmission rate for said data stream is decremented upon determining that a difference between a first time of a first data packet and a last time of a last data packet in said playout buffer drops below a dynamically computed decrement bandwidth (DEC_BW) threshold.

16. The client computer of claim **15** wherein said current number of packets remaining is represented by a current playtime which is set to a duetime of a last data packet in said playout buffer.

17. The client computer of claim **15** wherein the transmission rate is decremented upon determining a lossrate of the data packets exceeds a lossrate threshold.

18. The client computer of claim **15** wherein said data stream is a video data stream.

19. The client computer of claim **15** wherein the transmission rate is decremented upon determining an average

client computational rate of the client computer is less than the transmission rate.

20. The client computer of claim **15**, wherein the transmission rate for said data stream is decremented upon determining a difference between the current number of data packets and a previous number of data packets remaining in said playout buffer is less than a first constant.

21. A client computer useful in association with a server, the client computer coupled to the server via a computer network, the computer network providing a network connection with a variable bandwidth for transmitting a data stream from said server to said client computer, the data stream including a plurality of data packets, the client computer comprising:

a playout buffer for buffering the incoming data packets, wherein a transmission rate of said data stream is incremented upon determining a current number of data packets remaining in the playout buffer rises above a dynamically computed upper increment bandwidth (IN_BW) threshold and does not drop below a dynamically computed lower INC_BW threshold for at least a Time_Before_Increase.

22. The client computer of claim **21** wherein a Time_Buffer_Full indicates how long the current number of data packets remaining in the playout buffer has risen above the upper INC_BW threshold, wherein said Time_Buffer_Full is reset upon determining the current number of data packets remaining in the playout buffer rises above the upper INC_BW threshold and then drops below the lower INC_BW threshold within said Time_Before_Increase.

23. The client computer of claim **21** wherein said Time_Before_Increase is increased upon determining the transmission rate is reduced shortly after it was previously increased.

24. The client computer of claim **21** wherein said lower INC_BW threshold is intialized to a percentage of an ideal playout buffer size.

25. The client computer of claim **21** wherein the transmission rate is incremented upon determining an average computational rate of said client computer is greater than the transmission rate and upon determining the current number of data packets remaining in the playout buffer rises above the upper INC_BW threshold and does not drop below the lower INC_BW threshold for at least said Time_Before_Increase.

26. The client computer of claim **21** wherein said data stream is a video data stream.

27. The client computer of claim **21** wherein said current number of packets remaining is represented by a current playtime which is set to a duetime of a last data packet in said playout buffer.

28. The client computer of claim **21** wherein the transmission rate for said data stream is decremented upon determining the current number of data packets remaining in said playout buffer drops below a dynamically computed decrement bandwidth (DEC_BW) threshold, and upon determining a difference between the current number of data packets and a previous number of data packets remaining in said playout buffer is less than a first constant.

29. A computer-readable medium useful in association with a client computer coupled to a server via a computer network, the computer network providing a network connection with a variable bandwidth for transmitting a data stream from the server to the client computer, the data stream including a plurality of data packets, the client computer having a processor and a playout buffer for buffering said data packets from the server, the computer-readable medium comprising computer-readable code instructions configured to cause said client computer to execute:

initializing a data transmission rate for said data stream;

dynamically computing a decrement bandwidth (DEC_BW) threshold for the playout buffer; and

decrementing the transmission rate upon determining that a difference between a first time of a first data packet and a last time of a last data packet in said playout buffer drops below said dynamically computed DEC_BW threshold.

30. The computer-readable medium of claim **29** wherein said current number of packets remaining is represented by a current playtime which is set to a duetime of a last data packet in said playout buffer.

31. The computer-readable medium of claim **29** further comprising computer-readable code instructions configured to cause said client computer to execute:

determining a lossrate of the data packets; and

decrementing the transmission rate upon determining the lossrate exceeds a lossrate threshold.

32. The computer-readable medium of claim **29** wherein said data stream is a video data stream.

33. The computer-readable medium of claim **29** further comprising computer-readable code instructions configured to cause said client computer to execute:

determining an average client computational rate of said client computer; and

decrementing the transmission rate upon determining the average client computational rate is less than the transmission rate.

34. The computer readable medium of claim **29**, further comprising decrementing the transmission rate upon determining a difference between the current number of data packets and a previous number of data packets remaining in said playout buffer is less than a first constant.

35. A computer-readable medium useful in association with a client computer coupled to a server via a computer network, the computer network providing a network connection with a variable bandwidth for transmitting a data stream from the server to the client computer, the data stream including a plurality of data packets, the client computer having a processor and a playout buffer for buffering said data packets from the server, the computer-readable medium comprising computer-readable code instructions configured to cause said client computer to execute:

initializing a data transmission rate for said data stream;

initializing a Time_Before_Increase;

dynamically computing an upper increment bandwidth (INC_BW) threshold and a lower INC_BW threshold for said playout buffer; and

incrementing the transmission rate upon determining a current number of data packets remaining in the playout buffer rises above the upper INC_BW threshold and does not drop below the lower INC_BW threshold for at least said Time_Before_Increase.

36. The computer-readable medium of claim **35** wherein a Time_Buffer_Full indicates how long the current number of data packets remaining in the playout buffer has risen above the upper INC_BW threshold, the computer-readable medium further comprising computer-readable code instructions configured to cause said client computer to execute resetting the Time_Buffer_Full upon determining the current number of data packets remaining in the playout buffer

rises above the upper INC_BW threshold and then drops below the lower INC_BW threshold within said Time_Before_Increase.

37. The computer-readable medium of claim **35** further comprising computer-readable code instructions configured to cause said client computer to execute increasing said Time_Before_Increase upon determining the transmission rate is reduced shortly after it was previously increased.

38. The computer-readable medium of claim **35** wherein said lower INC_BW threshold is initialized to a percentage of an ideal playout buffer size.

39. The computer-readable medium of claim **35** further comprising computer-readable code instructions configured to cause said client computer to execute:

determining an average client computational rate of said client computer; and

incrementing the transmission rate upon determining the average client computational rate is greater than the transmission rate and upon determining the current number of data packets remaining in the playout buffer rises above the upper INC_BW threshold and does not drop below the lower INC_BW threshold for at least said Time_Before_Increase.

40. The computer-readable medium of claim **35** wherein said data stream is a video data stream.

41. The computer-readable medium of claim **35** wherein said current number of packets remaining is represented by a current playtime which is set to a duetime of a last data packet in said playout buffer.

42. The computer-readable medium of claim **35** further comprising computer-readable code instructions configured to cause said client computer to execute:

dynamically computing a decrement bandwidth (DEC_BW) threshold for the playout buffer; and

decrementing the transmission rate upon determining the current number of data packets in said playout buffer drops below said dynamically computed DEC_BW threshold, and upon determining a difference between the current number of data packets and a previous number of data packets remaining in said playout buffer is less than a first constant.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,292,834 B1
DATED : September 18, 2001
INVENTOR(S) : Ravi et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page,
Item [75], Inventors, change the fourth listed inventor's name to read
-- **David del Val** --.

Column 8,
Line 43, replace "716" with -- 714 --.

Column 9,
Line 16, replace "/Expected_" with -- /(Expected_ --.
Line 28, replace "=(High)" with -- =High) --.
Line 32, replace "stablize" with -- stabilize --.

Column 12,
Line 4, replace "posf" with -- "post" --.

Signed and Sealed this

First Day of April, 2003



JAMES E. ROGAN
Director of the United States Patent and Trademark Office