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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 September 2011 (29.09.2011)

PCT

(10) International Publication Number
WO 2011/119132 A2

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2010/000869

(22) International Filing Date:
24 March 2010 (24.03.2010)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **THOMSON LICENSING** [FR/FR]; 1-5 rue Jeanne d'Arc, F-92130 Issy Les Moulineaux (FR).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **PATEL, Bankim A** [US/US]; 39 Peterson Road, Hillsborough, New Jersey 08844 (US). **GUO, Yang** [CN/US]; 25 Wilson Way N, Princeton Junction, New Jersey 08550 (US).

(74) Agents: **SHEDD, Robert D.** et al.; Thomson Licensing LLC, 2 Independence Way, Suite #200, Princeton, New Jersey 08540 (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, RO, RS, RU, 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, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM,

[Continued on next page]

(54) Title: VARIABLE BIT RATE VIDEO STREAMING OVER PEER-TO-PEER NETWORKS

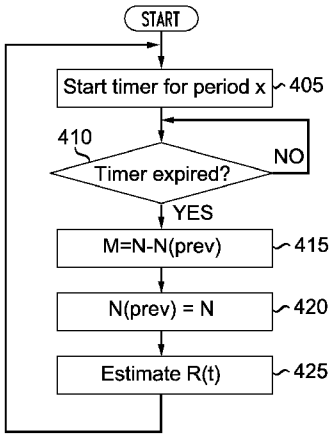

FIG. 4

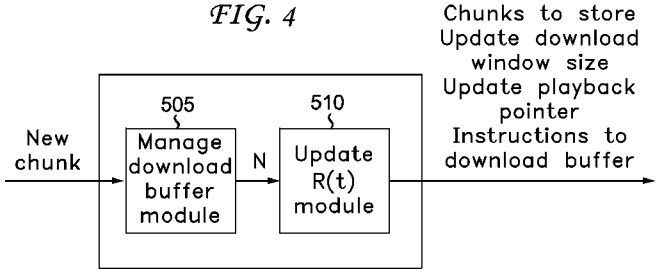

FIG. 5

(57) Abstract: A method and apparatus for managing a download buffer for variable bit rate streaming in a peer-to-peer network are described including receiving a chunk of content associated with a sequence number, determining a download window size, determining a sequence number distance responsive to the sequence number of the received chunk of content, comparing the sequence number distance to the download window size and a threshold value, performing one of storing the received chunk of content, dropping the received chunk of content, and updating the playback pointer and presenting chunks of content in the download buffer for rendering responsive to the comparison, determining a highest chunk sequence number responsive to sequence numbers of received chunks of content and estimating a variable bit rate responsive to the highest chunk sequence number.

WO 2011/119132 A2

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

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

VARIABLE BIT RATE VIDEO STREAMING OVER PEER-TO-PEER NETWORKS

FIELD OF THE INVENTION

The present invention relates to video streaming over peer-to-peer (P2P) networks and in particular, to the use of variable bit rate (VBR) coded video for streaming data over P2P networks.

BACKGROUND OF THE INVENTION

To support VBR video in P2P live streaming, the download buffer needs to be properly managed – the playback pointer needs to move forward at the rate of playback; and the buffer size has to be proportional to the current playback rate. Typically, the playback rate of VBR stream is not available to peers. The present invention proposes and describes practical techniques that enable peers to estimate the VBR playback rate and set the buffer size appropriately. The ability to estimate the VBR and set the buffer size accordingly make the streaming of VBR coded video over P2P networks feasible.

SUMMARY OF THE INVENTION

In P2P live streaming, video content is transmitted to peers via data sharing (exchanging) between and among participating peers. Typically, constant bit rate (CBR) video is used in P2P live streaming. VBR is more bandwidth efficient. How to support VBR video streaming over P2P networking has not been the subject of any significant research or study. A method and apparatus for supporting VBR coded video in P2P live streaming is described herein.

A method and apparatus for managing a download buffer for variable bit rate streaming in a peer-to-peer network are described including receiving a chunk of content associated with a sequence number, determining a download window size, determining a sequence number distance responsive to the sequence number of the received chunk of content, comparing the sequence number distance to the download window size and a threshold value, performing one of storing the received chunk of content, dropping the received chunk of content, and updating the playback pointer and presenting chunks of content in the download buffer for rendering responsive to the comparison, determining a highest chunk sequence number responsive to sequence numbers of received chunks of

content and estimating a variable bit rate responsive to the highest chunk sequence number.

BRIEF DESCRIPTION OF THE DRAWINGS

The present invention is best understood from the following detailed description when read in conjunction with the accompanying drawings. The drawings include the following figures briefly described below:

Fig. 1 is a schematic diagram of a VBR system.

Fig. 2 shows the structure of the download buffer of the present invention.

Fig. 3 is a flowchart of an exemplary embodiment of the manage download buffer portion of the P2P data engine of the present invention.

Fig. 4 is a flowchart of an exemplary embodiment of the rate determination portion of the P2P data engine of the present invention.

Fig. 5 is a block diagram of an exemplary embodiment of the P2P data engine of the present invention.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

Various P2P live streaming approaches have been proposed. All of these approaches share an architecture similar to the architecture depicted in Fig. 1. At the bottom is the P2P data engine layer 105 by which peers exchange the data chunks with each other in a P2P fashion. The missed chunks are fetched from other peers, while available chunks are uploaded to peers who request them. The received chunks are stored in the download buffer 110 of Fig. 1. Since the chunks are fetched out of order, the download buffer serves to assemble the chunks and present the chunks to the top layer video player 115 in the sequential order for rendering (display). Different P2P streaming approaches have different content sharing strategies. The present invention focuses on how to appropriately manage the download buffer so that VBR content can be streamed over P2P networks.

The structure of a download buffer is illustrated in Fig. 2. Video content is divided into equal length chunks. In Fig. 2 chunks which are presently stored in the download buffer are shaded while chunks which are missing appear white. The missing chunks that appear larger than an "equal size chunk" indicate multiple missing chunks. The chunks to

the right of the playback pointer 205 have been merged together (chunk division lines have been deleted) since these chunks have passed their playback deadline or are currently being rendered (displayed). Chunks are marked with sequence numbers in the order of their positions in the video content. The playback pointer 205 points to the last chunk that has not been delivered (presented) to the video player for rendering (displaying). All the chunks before (to the right of) the playback pointer 205 are past the playback deadline or are being consumed by the player. The data chunks after (to the left of) the playback pointer 205 but within the download window 210 will be downloaded from other peers. The end of the download window is indicated by the vertical arrow 215 below and to the far left of the download buffer. The download window is, therefore, between the playback pointer and then end of the download window as just described. The playback pointer 205 moves forward (to the left) as the time passes. In Fig. 2 time proceeds from right to left. The playback rate of VBR video varies over time and peers may not know the exact playback rate without getting out-of-band rate information. How to manage the download buffer is challenging in P2P streaming of VBR content.

One simple approach to address this issue is to use a fixed size download buffer, i.e., the buffer stores a fixed number of chunks. Let S be the buffer size (number of chunks) and $R(t)$ be the current playback rate (chunks/second). Then the total time required to playback the buffer (buffer playback time) is equal to $S/R(t)$. When the playback rate varies significantly as in VBR, the buffer playback time, $S/R(t)$, also varies dramatically. If the buffer playback time is too small, it may reduce the efficiency of P2P data engine. A mechanism is described herein that automatically estimates the VBR playback rate and properly manages the download buffer. Let P_1 be the chunk sequence number of the playback pointer, $R(t)$ be playback rate at time t, and T be the download window size in terms of playback time. Fig. 3 is a flowchart illustrating an exemplary method for peers to manage the download buffer in order to support VBR stream.

The peer performs basic initialization of the playback pointer, the playback rate, the time and the download window size to begin. The playback point (P1) is initialized (set) to zero and the current playback rate is set to a default initial value. Upon receiving a chunk with sequence number P, the peer first updates its download window size $W = R(t) T$, where is $R(t)$ is currently estimated playback rate, and T is download window size in

terms of playback time. *R(t)* is periodically updated by another thread, which will be described below and is depicted in Fig. 4. The sequence number distance, *D*, between this newly received packet and playback pointer is computed, $D=P-P_1$.

If *D* is greater than the download window size *W*, this chunk is beyond the current download window. At this point, download window needs to be moved forward, to make a room for this chunk. All the chunks between P_1 and *P-W* (excluding chunk *P-W*) will not be downloaded further and are presented to the video player immediately. The playback pointer is then set to *P-W*. When the playback pointer is moved the video player (and user) will experience jitter and the audio and video will jump during playback. The playback pointer is moved even though there are missing chunks of content. The missing packets are delayed and the download buffer cannot wait for the missing chunks due to the strict time constraints for delivery of live streaming content. (Note: P2P live streaming is operating under strict time constraints. Hence, to avoid delay during playback, the window size (W) should be configured based on worst case delay of chunk in the P2P network. A larger W gives very smooth playback, but it increases the startup delay).

If $0 < D \leq W$, the chunk falls into the current download window. The chunk will be stored into the download buffer.

Finally, if $D \leq 0$, this chunk lags behind the playback pointer and has missed its playback deadline. The packet is not useful and is dropped.

Referring again to Fig. 3, at 305 initialize by setting P1 = 0, t = 0 and R(t) = R(0) to a default value. At 310 receive a new chunk with sequence number P and also receive input (feedback) regarding the updated estimated variable bit rate. At 315 calculate (determine) the download window size, W, by setting W = R(t)T and set $D = P-P_1$. At 320 a test is performed to determine if D is greater than the current download window size (D>W). If D>W then at 325 instruct the download buffer to present chunks between P1 and P-W to the video player for rendering (displaying). The playback pointer is set to P-W at 330. Processing proceeds to 355. If D≤W then at 335 a test is performed to determine if D is both less than or equal to W and greater than 0. If D is both greater than 0 and less than or equal to W then at 340 the received chunk is stored in the buffer in its appropriate place (position) since the buffer is used to order the chunks of content. Processing proceeds to 355. If D is not both greater than 0 and less than or equal to W,

then at 345 a test is performed to determine if D is less than or equal to 0. If D is less than or equal to 0 then the chunk is dropped since this chunk lags behind the playback pointer and has missed its playback deadline. Processing proceeds to 355. At 355 a test is performed to determine if P is the largest sequence number. If P is the largest sequence number then set N = P at proceed to 310. If P is not the largest sequence number then proceed to 310. N is used in Fig. 4 to calculate (determine, estimate) R(t).

If the received chunk has highest sequence number seen so far, this sequence number is recorded using variable *N*. It will be used in estimating the current playback rate.

Playback rate information, *R(t)*, is required to manage the download buffer. One way to solve the problem of obtaining the playback rate is to have the server periodically transmit (send out) the rate information to peers. This solution, however, imposes extra signaling overhead on P2P networks. In the following, a self-learning playback rate estimation algorithm is proposed and described.

Peers estimate the current video streaming rate by measuring the number of packets that have been transmitted by the server. The playback rate is then updated periodically. Fig. 4 is a flowchart that illustrates an exemplary process used to estimate the playback rate.

Suppose the playback rate is re-estimated every x seconds. The timer is turned on at the beginning. At the end of every period (epoch), i.e., every x seconds, the number of chunks newly transmitted by server, M, is estimated by $N\text{-}N(prev)$, where N is the highest chunk sequence number observed so far and $N(prev)$ is the highest chunk sequence number observed in the previous round (period). Then $R(t)$ is set to be $R(t)=M/x$. An exemplary time period, x, is one second.

There are several variations to the playback rate estimation. For instance, the period of x seconds may be too short for estimation. One way to improve the estimation is to average the rate over longer period of time.

Assuming that a history of the last K playback rates $\{R(i)\}^{K}_{i=1}$, where, $R(1)$ is the most recent rate is maintained:

$$R(t) = (R(1) + R(2) + \ldots + R(K)) / K.$$

Another possible extension is using weighted average.

$R(t) = (w1*R(1) + w2*R(t2) + \ldots + w_K *R(K))$,

where $w1, w2, \ldots, wK$ are weight coefficients, $w1 + w2 + \ldots + wK = 1$, and w1 > w2 > w3 > … > wK.

Referring again to Fig. 4, at 405 a timer is started for the period x. The timer may be a timer that counts up or that counts down. A test is performed at 410 to determine if the timer has expired. If the timer has not expired then the timer stays in a tight loop continually checking for timer expiration. If the timer has expired then at 415, M is set equal to N less the previous value of N denoted as $N(prev)$. At 420 $N(prev)$ is set to the value of N and at 425 $R(t)$ is estimated by any one of the methods described above.

Fig. 5 is a block diagram of an exemplary embodiment of the P2P data engine of the present invention. The P2P data engine may be implemented in hardware, software, firmware, special purpose processors or any combination thereof. This includes, but is not limited to, application specific integrated circuits (ASICs), reduced instruction set computers (RISCs), and/or field programmable gate arrays (FPGAs). Preferably, the present invention is implemented as a combination of hardware and software. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage device. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. The present invention may be implemented in more or fewer modules (components) tan indicated in Fig. 5 and described herein. There is a module to manage the download buffer 505 in a VBR streaming environment. This module accepts a new chunk of content and performs the functions described in Fig. 3 including outputting a new value of N to the rate update module 510. That is, in one embodiment of the present invention the manage download buffer management module would include a means for receiving a chunk of content associated with a sequence number, a means for determining a download window size, a means for determining a sequence number distance responsive to the sequence number of the received chunk of content, a means for comparing the sequence number distance to the download window size and a threshold value, and a means for performing one of storing the received chunk of content, dropping the received chunk of content, and updating the playback pointer and presenting chunks of content in the download buffer

for rendering responsive to results of the means for comparing. The manage download buffer module also accepts feedback from the rate update module. The rate update module accepts a value of N from the buffer management module and estimates a new variable bit rate used by the buffer management module to manage the download buffer. The rate update module of one embodiment of the present invention includes a means for determining a highest chunk sequence number responsive to sequence numbers of received chunks of content and a means for estimating a variable bit rate responsive to the highest chunk sequence number. The modules of the P2P data engine output chunks of content to store in the download buffer, update the download window size, update the playback pointer and provide instructions to the download buffer regarding when to forward the chunks of content stored therein to the video player to render. The use of the term video player should not be taken literally and may include any device which can display and/or render the video stream including a TV, a display monitor or any other equivalent device.

It is to be understood that the present invention may be implemented in various forms of hardware, software, firmware, special purpose processors, or a combination thereof. Preferably, the present invention is implemented as a combination of hardware and software. Moreover, the software is preferably implemented as an application program tangibly embodied on a program storage device. The application program may be uploaded to, and executed by, a machine comprising any suitable architecture. Preferably, the machine is implemented on a computer platform having hardware such as one or more central processing units (CPU), a random access memory (RAM), and input/output (I/O) interface(s). The computer platform also includes an operating system and microinstruction code. The various processes and functions described herein may either be part of the microinstruction code or part of the application program (or a combination thereof), which is executed via the operating system. In addition, various other peripheral devices may be connected to the computer platform such as an additional data storage device and a printing device.

It is to be further understood that, because some of the constituent system components and method steps depicted in the accompanying figures are preferably implemented in software, the actual connections between the system components (or the process steps) may differ depending upon the manner in which the present invention is

programmed. Given the teachings herein, one of ordinary skill in the related art will be able to contemplate these and similar implementations or configurations of the present invention.

CLAIMS:

1. A method for managing a download buffer for variable bit rate streaming in a peer-to-peer network, said method comprising:

receiving a chunk of content associated with a sequence number;

determining a download window size;

determining a sequence number distance responsive to said sequence number of said received chunk of content;

comparing said sequence number distance to said download window size and a threshold value;

performing one of storing said received chunk of content, dropping said received chunk of content, and updating said playback pointer and presenting chunks of content in said download buffer for rendering responsive to said comparison;

determining a highest chunk sequence number responsive to sequence numbers of received chunks of content; and

estimating a variable bit rate responsive to said highest chunk sequence number.

2. The method according to claim 1, wherein said chunks of content presented for rendering are between said playback pointer and the received chunk of content less the download window size.

3. The method according to claim 1, further comprising:

initializing said layback pointer;

initializing a time; and

setting said variable bit rate to an initial value.

4. An apparatus for managing a download buffer for variable bit rate streaming in a peer-to-peer network, comprising:

means for receiving a chunk of content associated with a sequence number;

means for determining a download window size;

means for determining a sequence number distance responsive to said sequence number of said received chunk of content;

means for comparing said sequence number distance to said download window size and a threshold value;

means for performing one of storing said received chunk of content, dropping said received chunk of content, and updating said playback pointer and presenting chunks of

content in said download buffer for rendering responsive to results of said means for comparing;

means for determining a highest chunk sequence number responsive to sequence numbers of received chunks of content; and

means for estimating a variable bit rate responsive to said highest chunk sequence number.

5. The apparatus according to claim 4, wherein said chunks of content presented for rendering are between said playback pointer and the received chunk of content less the download window size.

6. The apparatus according to claim 4, further comprising:

means for initializing said layback pointer;

means for initializing a time; and

means for setting said variable bit rate to an initial value.

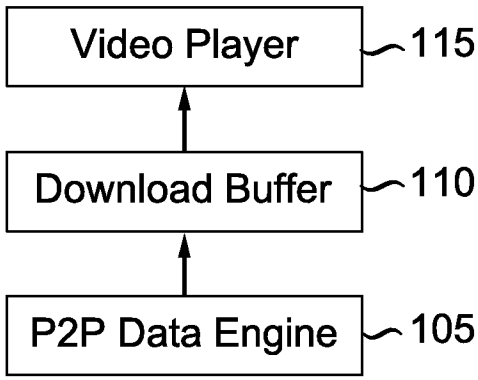
Video Player
115
Download Buffer
110
P2P Data Engine
105

FIG. 1

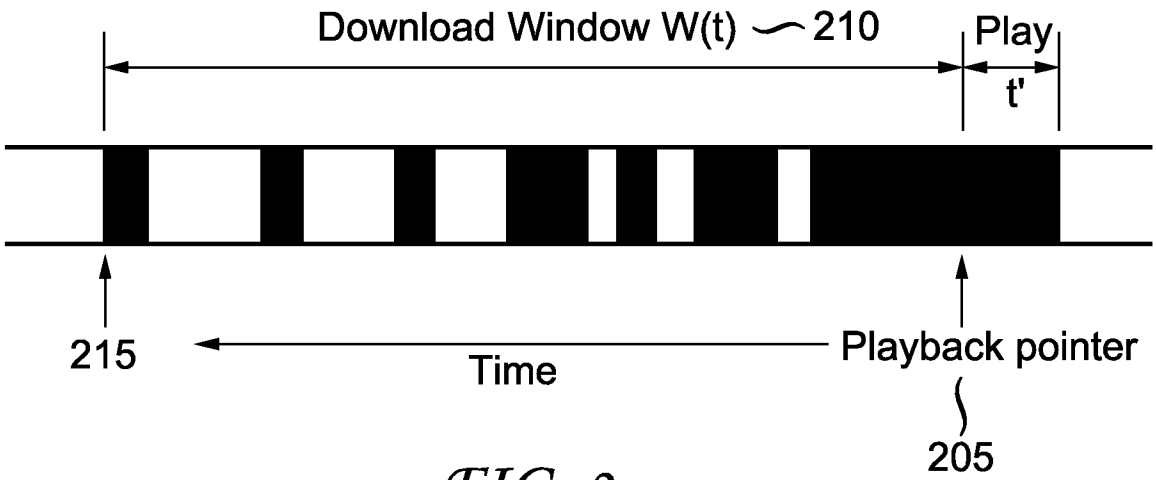
Download Window W(t)
210
Play
t'
215
Time
Playback pointer
205

FIG. 2

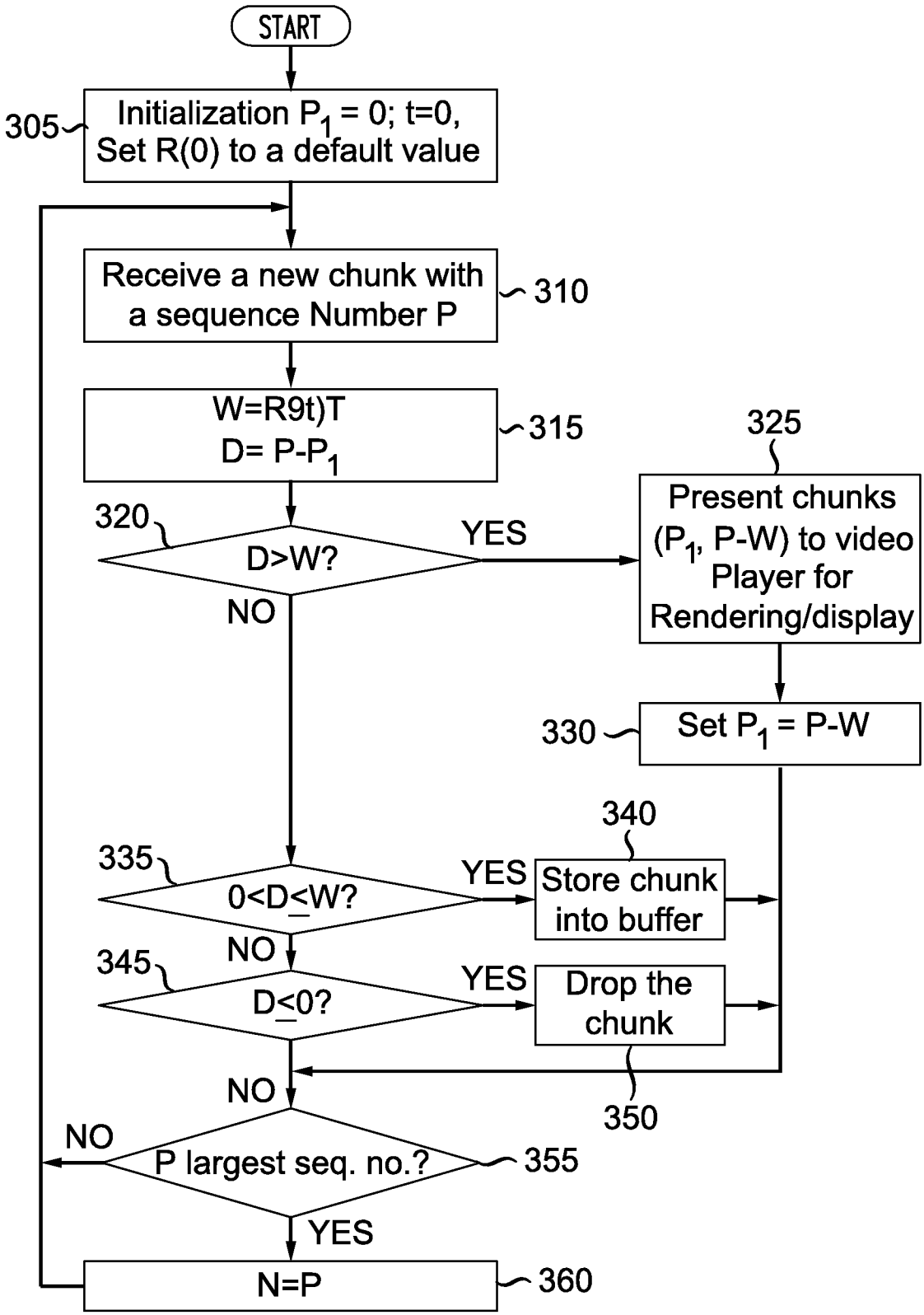
START
305
Initialization P_1 = 0; t=0,
Set R(0) to a default value
Receive a new chunk with
a sequence Number P
310
W=R9t)T
D= P-P_1
315
320
D>W?
YES
NO
325
Present chunks
(P_1, P-W) to video
Player for
Rendering/display
330
Set P_1 = P-W
335
0<D≤W?
YES
340
Store chunk
into buffer
NO
345
D≤0?
YES
Drop the
chunk
NO
350
NO
P largest seq. no.?
355
YES
N=P
360

FIG. 3

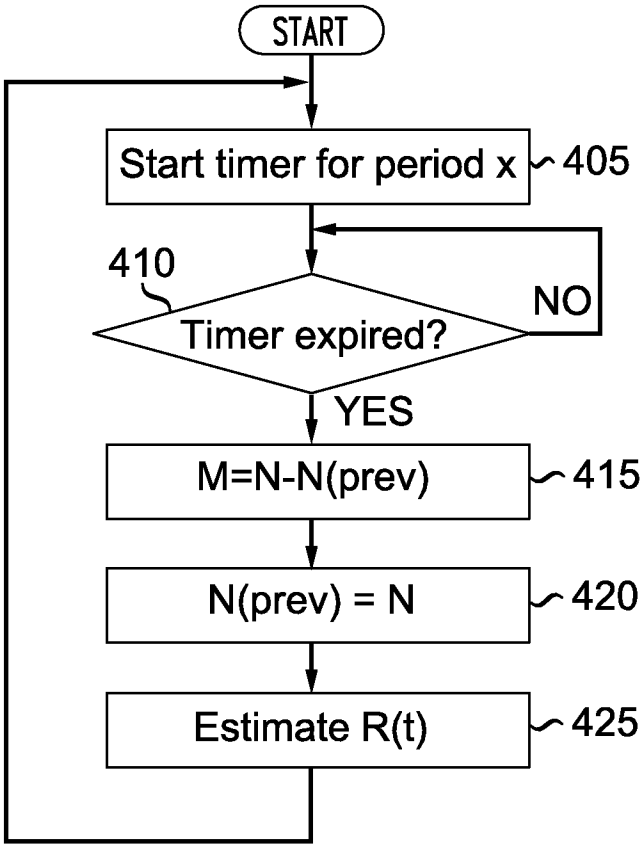
START
Start timer for period x
405
410
Timer expired?
NO
YES
M=N-N(prev)
415
N(prev) = N
420
Estimate R(t)
425

FIG. 4

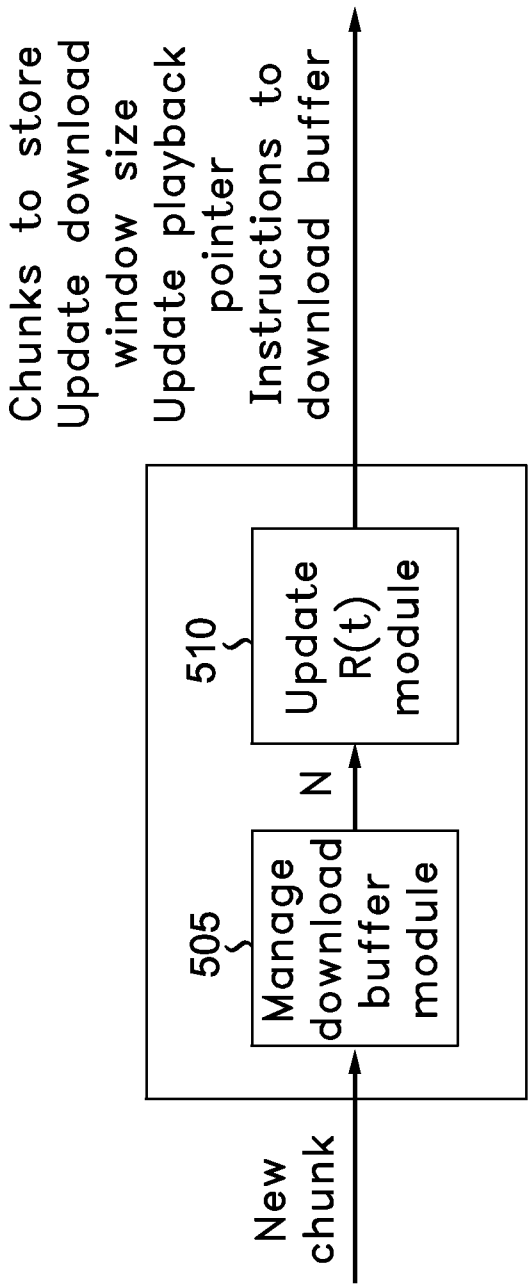
Chunks to store
Update download window size
Update playback pointer
Instructions to download buffer
510
Update R(t) module
N
505
Manage download buffer module
New chunk

FIG. 5