

packets to a negligibly small number, e.g., 0.1 packets for every retransmission interval is defined by m . This implies that:

$$P_{loss}^m * R_p \leq \frac{0.1}{x}, \text{ or that } m \geq \frac{\log\left(\frac{0.1}{R_p * x}\right)}{\log P_{loss}}$$

Alternatively, if s is the packet size, the above equation may be expressed as:

$$m \geq \frac{\log\left(\frac{0.1 * s}{e_r * x}\right)}{\log P_{loss}}$$

The minimum required buffer size may be defined as equal to $m * x$.

Consequently, the size constraint of the backup buffer 26 (Figure 1) is

$$b_{backup} \geq m * x \geq x * \frac{\log\left(\frac{0.1 * s}{e_r * x}\right)}{\log P_{loss}} \quad (\text{Equation 3})$$

The detailed operational method underlying the invention is captured through a protocol called "Streaming Congestion Protocol" (SCP). In the section below the SCP sequences during different phases of streaming are described. Figures 6 and 7 show the detailed operational sequence at the streaming server process 16 (Figure 3) and the streaming player process 20 (Figure 3) during normal streaming of the streaming packets 70 (Figure 3). The operational sequences are described with the help of state diagrams. A state diagram shows various states and transitions between states. Each state captures a specific operational state, and the transitions between states capture the events that cause the operational states to change. Figure 7 shows the state diagram of events at the streaming server process 16 (Figure 1). Figure 8 shows the corresponding state diagram at the streaming player process 20.

Processing during and after congestion

Please refer to Figure 9, by default, the streaming player process 20 is in a normal state 100. The player process 20 may keep a track of the packet loss of a received stream at a periodic interval that may be user determined. If the amount of loss exceeds a

threshold, e.g., Threshold1, across a time-window of some number of seconds which may also be user determined, e.g., TW1, the player process 20 moves into the Likely Congestion state 102. At this point, the streaming player process 16 (Figure 1) launches congestion probes to determine whether the congestion symptoms are verified. These probes form a part of a test called confirm congestion test. If the test indicates that congestion is confirmed, the process moves to a Congestion Confirmed state 104. Immediately, the streaming player process 20 contacts its server counterpart and informs it about the new state change along with some other relevant information.

Note that the transition from a Likely Congested state 102 to Congestion Confirmed State 104 can happen in two other ways. First, the player process 20 may actually receive a Remote Congestion Confirmed report in state 106 from its server process 16 (Figure 1) indicating that some other player process 20 on a related network path is in the Congestion Confirmed state. If this happens, the player immediately moves to the Congestion Confirmed State 104 itself. If the player process 20 has already started conducting congestion probes before this notification, the player process 20 may halt them at the instant it receives the notification. Alternatively, the player process 20 may receive a Local Congestion Confirmed report in state 102 reporting that some other local player process 20 on the same streaming player module has verified congestion. If the Remote or local Congestion Confirmed reports have verified congestion within a past time-window not exceeding the time defined as Tcc, the player process 20 moves to a confirmed congestion state 104.

Once the streaming player process 20 (Figure 1) enters the congestion confirmed state 104, it periodically monitors the loss of the received stream of streaming packets (Figure 3). If the packet loss is less than a second user defined threshold, e.g., Threshold2, across a second user defined time-window, e.g., TW2 seconds, the player process 20 enters the likely end-of-congestion (EOC) state 106. At this point, the player process 20 launches an EOC probe test that either confirms or invalidates the EOC decision. If the EOC probe test confirms congestion, the streaming player process 20 enters the EOC state 108 and sends an EOC report 110 to its server process 16 (Figure 1) counterpart. With this transition, the process re-enters the normal state 100. Note that

during transitions from the EOC state 110 back to the Normal State 100, the streaming player process 20 also needs to continually process the arriving packets in step 120 (Figure 7), check and request for retransmission of missing packets in step 122 (Figure 7), and forward packets to the streaming player for playback in step 124 (Figure 7).

5 These activities are indicated in Figure 7 and are not shown in Figure 9 for purposes of clarity.

Figure 10 shows the SCP state-diagram of the server process 16. After initialization, the server process 16 enters the normal state 130. If the server process 16 receives a report from its streaming player process 20 (Figure 1) indicating congestion, the server moves to the Confirmed Congestion or Confirmed Congestion State 132. At this point, the server process 16 shares the Confirmed Congestion status information with other server processes 16 running on the same computing device 14 (Figure 1) in the Propagate Confirmed Congestion state 134, and potentially also from other computing device 14 (Figure 1) through interaction with the service manager 12 (Figure 1) via the network 22 (Figure 1). The idea behind this is to have selected server processes report the congestion to their player process so that these player processes can avoid doing the Confirmed Congestion probing test. This cuts down on the congestion reaction time and significantly cuts down the probing traffic. In general, each server process that receives the Confirmed Congestion status in the Receive the Confirmed Congestion status state 140 from other server processes 16, first determines whether the Confirmed Congestion status is relevant to its target player process 20 (Figure 1). If this is the case, the server processes notify their player process 20 (Figure 1). Notification is done through a special signaling scheme involving reliable transport protocol (RTP) messages, as discussed in Schulzrinne, Casner, Frederick, Jacobson, "RTP: A Transport Protocol for Real-time Applications", RFC 1889, Internet Engineering Task Force.

10
15
20
25

The streaming server process 16 will then make transitions from the Confirmed Congestion State 132. Starting with the Confirmed Congestion state 132, the server process 16 transmits streaming packets 70 (Figure 3) at a constant bit-rate. The server process 16 moves to the Recovery Scheduling State 136 as soon as it receives an End-of-congestion or EOC report from the player process 20 (Figure 1) targeted for the reception of

30

its content. Thereafter the server process 16 changes its mode of transmission, using recovery scheduling to drain its backlogged output buffer 24 (Figure 1) in an intelligent way. As soon as the backlog is over, the server process 16 moves back to the normal state 130.

5 While in the Normal State 130 the server process 16 may receive and filter information from other server processes 16 by entering the Receive Confirmed Congestion State 140. It may also notify the player processes 20 (Figure 1) about the confirmed congestion status through the in-band RTP signaling, by entering the Confirmed Congestion Notification state 138 from where the process 16 will return to the
10 Normal state 130.

The server process 16 also needs to continually process the departing packets in step 150 (Figure 8), check and process the requests for retransmission of missing packets in step 152 (Figure 8), and backup packets to the backup buffer 26 (Figure 1) for playback. These activities are indicated in Figure 8 and are not shown in Figure 10 for
15 purposes of clarity.

Recovery scheduling algorithm

Supposing that the server process 16 (Figure 1) streams of streaming packets 70 (Figure 3) are classified into K number of priority classes. Class 1 being the highest
20 priority and class "K" being the lowest priority. Each priority class "j" is associated with a weighting function α^j , where $0 < \alpha^K < \alpha^{K-1} < \dots < \alpha^1 = 1$. Considering the server process 16 (Figure 1) with priority equaling to "j", the rule for transmission rate e_i of this process is:

$$e_i = \min(\alpha^j \times SR_{min}, \alpha^j \times \frac{B - \lambda_{other}}{\sum_{i=1}^K \alpha^i n^i}) \quad \text{(Equation 4)}$$

25 In equation 4, all the terms are measured at the service bottleneck hop between the computing device 14 executing the server process 16 and the computing device 18 executing the player process 16. B is the bottleneck link capacity at the bottleneck hop.

λ_{other} is the aggregate packet rate carried at the hop from traffic not utilizing the inventive method. This rate can be estimated by obtaining the aggregate packet rate of the streaming packets 70 (Figure 3) of the inventive method λ_{SME} from the service manager, and using the fact that $\lambda_{other} = B \times \rho - \lambda_{SME}$, where $\rho = 1 - \frac{SR_{min}}{B}$ is the utilization of the bottleneck hop. The parameter n^i is defined as the number of priority "i" streams carried at the bottleneck hop. The hop is a section of a network between two network-computing devices such as routers. This parameter may be obtained from the service manager 12 (Figure 1). The logic behind this equation is as follows. The first term indicates that the new transmission rate should be set equal to the minimum service rate along the path, in order to recover from the backlog. The second term takes into account the result of increasing the transmission rate on the utilization at the bottleneck hop. In other words, since the maximum utilization can be 100%, each computing device 14 executing server process 16 must carefully weigh the effects of increasing the transmission rate. The best rate is the maximum rate that does not congest the bottleneck bandwidth during recovery. Hence the min operator. Finally, the priorities within the equation 4 provide for differentiated streaming.

Congestion Estimators

Earlier on, we described the congestion estimation tests as part of the SCP algorithm. In principle, congestion can be estimated in a variety of ways, and the invention is not restricted to any one particular method of congestion estimation. A particular embodiment of the invention for estimating congestion based on a mix of passive and active probes will now be described. This method comprises a set of rules configured around the results of the probes. A specific congestion test e.g., an end of congestion test or a confirmed congestion test mentioned in earlier sections, combines these rules in a particular way.

The following are some of a variety of methods that may be used to compute the bottleneck bandwidth on a path between the streaming player 20 (Figure 1) and the

streaming server 16 (Figure 1). These methods are shown here for exemplary purposes only and not to indicate that only these methods must be used with the present invention.

Capacity Probe

This method is described in detail in Van Jacobson, "pathchar - A tool to infer characteristics of Internet paths", MSRI, April 21, 1997; and R. L. Carter and M.E. Crovella, "Measuring Bottleneck Link Speed in Packet-switched Networks", TR-96-006, Boston University Computer Science Department, March 15, 1996. This method is initiated by the player process 20 (Figure 1) at various stages of a streaming session. The steps of this exemplary method are as follows:

1. Do a traceroute and get hop identities marked by IP addresses.
2.
 - a) For every hop from one onwards, send a series of ICMP timestamp requests (type 13) from the player to the hop. Packet sizes of queries are changed in steps of 1024, 512, 256, 128, and 64, where each queries are sent out with a spacing of 3 seconds between them.
 - b) For every hop i between the player and the server, the player must record the ICMP timestamp response (type 14) and compute the time difference between the hop i transmit time t_i and the player receive instant t_1 .
 - c) Denote the time difference by $\tau_{i1} = t_i - t_1$.
 - d) Take four measurements for timestamp differences by varying packet size through five values. Discard the minimum and the maximum, and take average of the remaining two. Let s' and s'' denote the packet sizes of two selected queries to hop i . Let $\tau_{i1'}$ and $\tau_{i1''}$ denote the measurements for the respective packet sizes.
 - e) Calculate $X_{i1} = \frac{\tau_{i1'} - \tau_{i1''}}{s' - s''}$

$$B_{i,i-1} = \frac{1}{X_{i1} - \left(\sum_{k=1}^{i-2} \frac{1}{B_{i-k,i-k-1}} \right)}$$

It can be proved that link capacity can be computed recursively.

The path bottleneck bandwidth $B_{min} = \min_i B_{i,i-1}$. (Equation 5)

5 Service Rate Probe

The "Service Rate probe" computes the approximate service rate of transferring packets between the server and the player, as discussed in Van Jacobson, "pathchar - A tool to infer characteristics of Internet paths", MSRI, April 21, 1997; and R. L. Carter and M.E. Crovella, "Measuring Bottleneck Link Speed in Packet-switched Networks", TR-96-006, Boston University Computer Science Department, March 15, 1996. The minimum service rate is the least service rate on the path, while the average service rate is the "expected" service rate on the path. . The steps of this exemplary method are as follows:

1. Obtain the hop list through traceroute. Suppose there are n-1 hops between the player and the server.
2. Periodically, the player initiates a cycle. In each cycle, the player sends n-1 queries, each spaced by ¼ second. The player forms timestamp differences between query responses from consecutive hops. The results are sent back to the server.
3. The server uses results from several cycles along with the link capacities to calculate the service rate.

- a) First, the server calculates $\tau_{m,m-1}^{min} = \min(\tau_{m,1} - \tau_{m-1,1})$ where the min operator works over all collected cycle results. We approximate $\tau_{m,m-1}^{min}$ to be the base-line delay between node m and node m-1.
- b) Denote the maximum queuing delay between node m and node m-1 by $q_{m,m-1}^{max}$, where node is a computing device, e.g., a computer or a router connected to the network.

c) Then, $q_{m,m-1}^{max} = \max(\tau_{m,m-1} - \tau_{m,m-1}^{min})$

d) Denote the average queuing delay between node m and node m-1 by $q_{m,m-1}^{avg}$.

e) Then, $q_{m,m-1}^{avg} = \text{avg}(\tau_{m,m-1} - \tau_{m,m-1}^{min})$

5 Finally, the minimum service rate is:

$$SR_{m,m-1}^{min} = \frac{s}{\frac{s}{B_{m,m-1}} + q_{m,m-1}^{max}}$$

and the average service rate is:

$$SR_{m,m-1}^{avg} = \frac{s}{\frac{s}{B_{m,m-1}} + q_{m,m-1}^{avg}} \quad (\text{Equation 6})$$

Congestion Detection Rule (CC Test)

A robust but responsive congestion detection mechanism is an essential part of the
 10 inventive method. It is important to respond quickly to congestion; this can reduce the
 congestion period and hence make it easier for the recovery scheduling to do its job in a
 shorter period of time. Ultimately this can allow the method to work with a reduced
 buffer size and hence a reduced connect lag. It is equally important to robustly
 distinguish a sustained congestion condition. False detection can lead to reduced
 15 throughput and performance. Recall that the inventive method treats sustained
 congestion by reducing the throughput of the newer packets, so that the throughput of
 older retransmitted packets is maximized.

The present invention uses the following congestion detection rule:

1. Packet losses exceeding a threshold for a consecutive time-window of duration T_s .
 20 T_s is set to 10 seconds by default.
2. If condition 1 is triggered, three short service rate probes are sent, spaced 2
 seconds apart, from the player to the SR_{min} hop. Responses are averaged to form

$$SR_{avg}$$

$$\rho = 1 - \frac{SR_{avg}}{B}$$

The player process 20 (Figure 1) computes the expected utilization ρ . If condition 1 is satisfied and the utilization from condition 2 is more than a specified utilization threshold, e.g., 95%, the player process 20 (Figure 1) declares sustained congestion (Confirmed Congestion), and asks the server to enter the constant bit-rate emulation mode. An alternative method may consists of a simple rule where if packet losses exceed a threshold for N_1 consecutive sampling time-windows, the player process 20 (Figure 1) declares sustained congestion and asks the server to enter the constant bit-rate emulation mode. By default, N_1 is set to 1, and time-window is set equal to 1 second.

10

End of Congestion Rule (EOC Test)

The end of congestion rule is analogous to the congestion detection rule:

1. No packet losses for a consecutive time-window of duration T_e . T_e is set equal to 10 seconds by default.
- 15 2. If condition 1 is triggered, three short service rate probes are sent, spaced 2 seconds apart, from the player to the SR_{min} hop. Responses are averaged to form SR_{avg} .

The player computes the expected utilization $\rho = 1 - \frac{SR_{avg}}{B}$.

If condition 1 is satisfied and the utilization from condition 2 is less than a specified threshold, e.g., 90%, the player declares sustained congestion to be over, and asks the server to transition to the state of recovery scheduling. An alternative method may consists of a simple rule where if in a state of sustained congestion, no packet losses are manifest for N_2 consecutive sampling time-windows, the player declares sustained congestion to be over, and asks the server to transition to the state of recovery scheduling. By default, N_2 may be set equal to five, and the sampling time-window is set equal to 1 second.

25

Thus, an apparatus and method for predictable and differentiated delivery of multimedia streaming on the Internet is provided. One skilled in the art will appreciate that the present invention can be carried out in other ways and practiced by other than the described embodiments, yet not departing from the spirit and essential characteristics depicted herein. The present embodiments therefore should be considered in all respects
5 as illustrative, and the present invention is limited only by the claims that follow.

CLAIMS

Having thus described our invention, what we claim as new, and desire to secure by Letters Patent is:

5

1. A method of streaming of multimedia audio and video packets from at least one sending computing device to an at least one receiving computing device via a network, said method comprising the steps of:

- 10 a) at a start of said streaming communicating configuration information from a service manager via the network to both an at least one sending computing device for use by an at least one server process and an at least one receiving computing device for use by an at least one player process, said at least one sending computing device comprising an at least one retransmit buffer, an at least one output buffer, an at least
15 one backup buffer and said at least one server process, said at least one receiving computing device, comprising an at least one player buffer and an at least one player process;
- b) queuing same packets in the at least one output buffer and to the at least one backup buffer;
- 20 c) determining if there are packets in the at least one output buffer and transmitting the packets from the at least one output buffer to the at least one receiving computing device via the network;
- d) receiving the packets on the at least one receiving device and storing them in the at least one player buffer;
- 25 e) determining if there are omissions in the packets stored in the at least one player buffer and forwarding retransmission requests to the at least one sending computing device and storing them in the at least one retransmit buffer; and
- f) determining if there are said retransmit requests in the at least one retransmit buffer and transmitting the packets from the at least one backup buffer to the at least one
30 receiving computing device via the network.

2. The method of claim 1 wherein, the network is Internet.

3. The method of claim 2 wherein, a size of the at least one player buffer is
 5 $\geq c * x * e_r * P_{loss}$, where c is a number of sampling intervals in a congestion period, x is
 a sampling or retransmission interval, $c * x$ is a worst-case duration of congestion, e_r is
 an encoding bit-rate of a stream of the packets and P_{loss} is a worst case of loss of the
 packets.

10 4. The method of claim 3 wherein, a size of the at least one output buffer is
 $\geq c * x * e_r * P_{loss}$.

5. The method of claim 4 wherein, a size of the at least one backup buffer is
 $\geq m * x \geq x * \frac{\log\left(\frac{0.1 * s}{e_r * x}\right)}{\log P_{loss}}$, where m is a number of retransmits that will reduce the
 15 loss rate for older packets to a negligibly small number, and $m * x$ is a minimum required
 size of the at least one player buffer.

6. A method of detecting and reacting to a network congestion which interferes
 with streaming of multimedia audio and video packets from at least one sending
 20 computing device to an at least one receiving computing device via a network, said
 method comprising the steps of:

a) communicating configuration information including network parameters from a
 service manager via the network to the at least one sending computing device and to
 the at least one receiving computing device, the at least one sending computing device
 25 comprising at least one server process, the at least one receiving computing device
 comprising at least one player process;

b) confirming the network congestion by transmitting and retracting network probes;

- c) executing at least one congestion control technique to limit degradation in the streaming of the packets;
- d) detecting an end of the network congestion by transmitting and retracting said network probes;
- 5 e) configuring a rate of streaming of the packets according to said network parameters

7. The method of claim 6, wherein said at least one congestion control technique is the constant bit-rate emulation.

- 10 8. The method of claim 7, wherein said at least one congestion control technique only reduces a transmission rate of the packets by the amount of incurred loss of the packets.

- 15 9. The method of claim 7, further comprising the steps of:
detecting an end of congestion;
computing available bandwidth in the network; and
transmitting packets according to said bandwidth.

- 20 10. The method of claim 9, wherein some of the at least one player process are differentiated and protected from effects of said network congestion to a greater than others by receiving a stream backlog sooner.

- 25 11. The method of claim 10, wherein some of the at least one player process are differentiated and protected from effects of said network congestion to a greater than others by receiving a stream backlog sooner according to a rule for transmission rate of the packets

$$e_i = \min(\alpha^j \times SR_{min}, \alpha^j \times \frac{B - \lambda_{other}}{\sum_{i=1}^{i=K} \alpha^i n^i}), \text{ where } e_i \text{ indicates said transmission rate of packets,}$$

α^j is a weighting function, for class $\{1 \dots K\}$.

12. The method of claim 11, wherein said detecting of said end of the network congestion is performed according to the expected utilization $\rho = 1 - \frac{SR_{avg}}{B}$, where SR_{avg} is the average rate at which the packets are transferred from the at least one server process to the at least one player process and B is a bottleneck link capacity at a bottleneck hop.

13. The method of claim 12, wherein said network probes compute an approximate service rate of transferring the packets from said at least one server process to said at least one player process according to the following equation:

$$SR_{m,m-1}^{min} = \frac{S}{\frac{S}{B_{m,m-1}} + q_{m,m-1}^{max}}, \quad SR_{m,m-1}^{avg} = \frac{S}{\frac{S}{B_{m,m-1}} + q_{m,m-1}^{avg}}, \quad \text{where } SR_{avg} \text{ and } SR_{min} \text{ are}$$

averaged and minimum service rate, $q_{m,m-1}^{max}$ is the maximum queuing delay between a node m and a node $m-1$.

14. A method of guaranteed delivery of streaming multimedia audio and video packets from at least one sending computing device to an at least one receiving computing device via a network, said method comprising the steps of:

- a) continuously processing arriving packets;
- b) keeping track of loss of the packets at the at least one receiving computing device, said at least one receiving computing device comprising one or more player processes;
- c) determining if said loss exceeds a first threshold during a first interval;
- d) verifying congestion symptoms;
- e) communicating said verified congestion symptoms to the at least one sending computing device, said at least one sending computing device comprising one or more server processes;

- f) monitoring said loss and determining if said loss exceeds a second threshold during a second interval to determine an end of congestion; and
- g) verifying congestion symptoms.

5 15. The method of claim 14, wherein said step of keeping track is performed at user define intervals.

 16. The method of claim 15, wherein said first threshold, said second threshold, said first interval, and said second interval are user defined.

10

 17. The method of claim 16, wherein said congestion symptoms are verified by receiving a remote congestion confirmed report.

 18. The method of claim 17, wherein said congestion symptoms are verified by
15 performing a congestion probe.

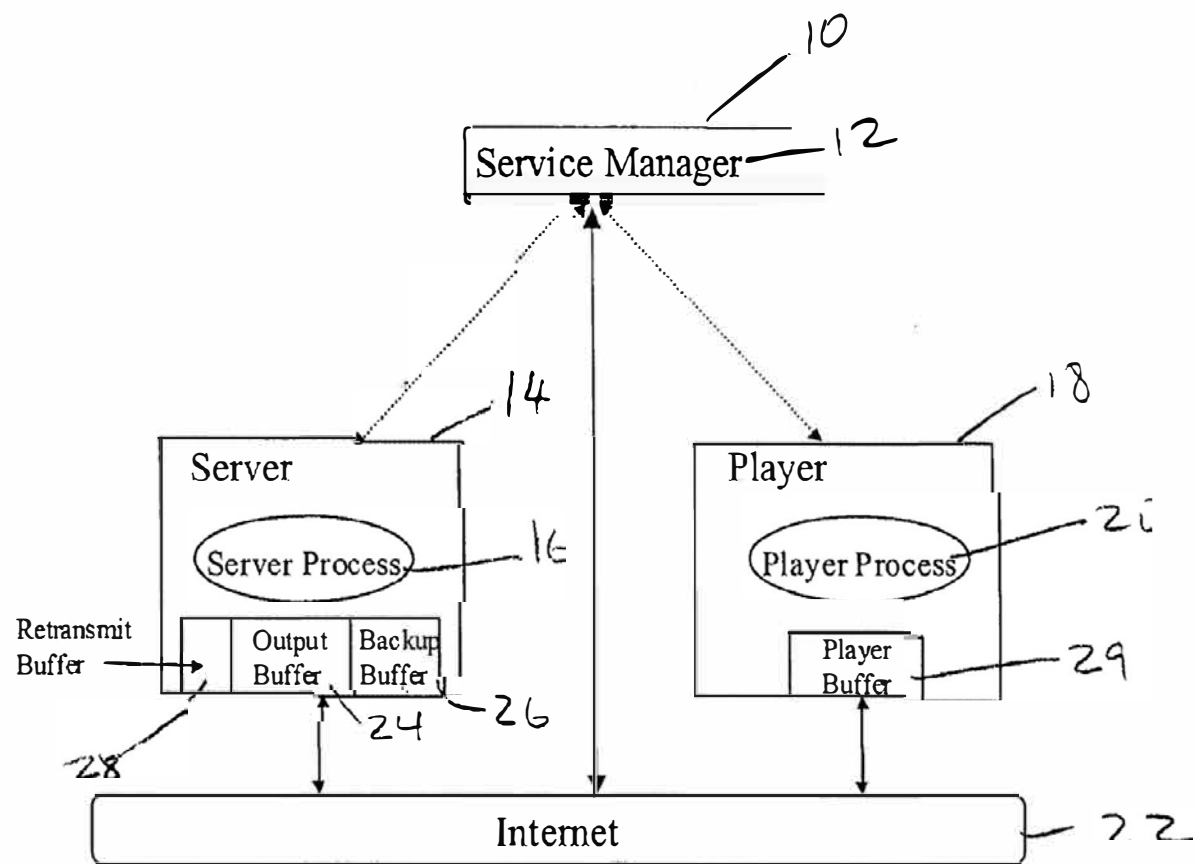


Figure 1

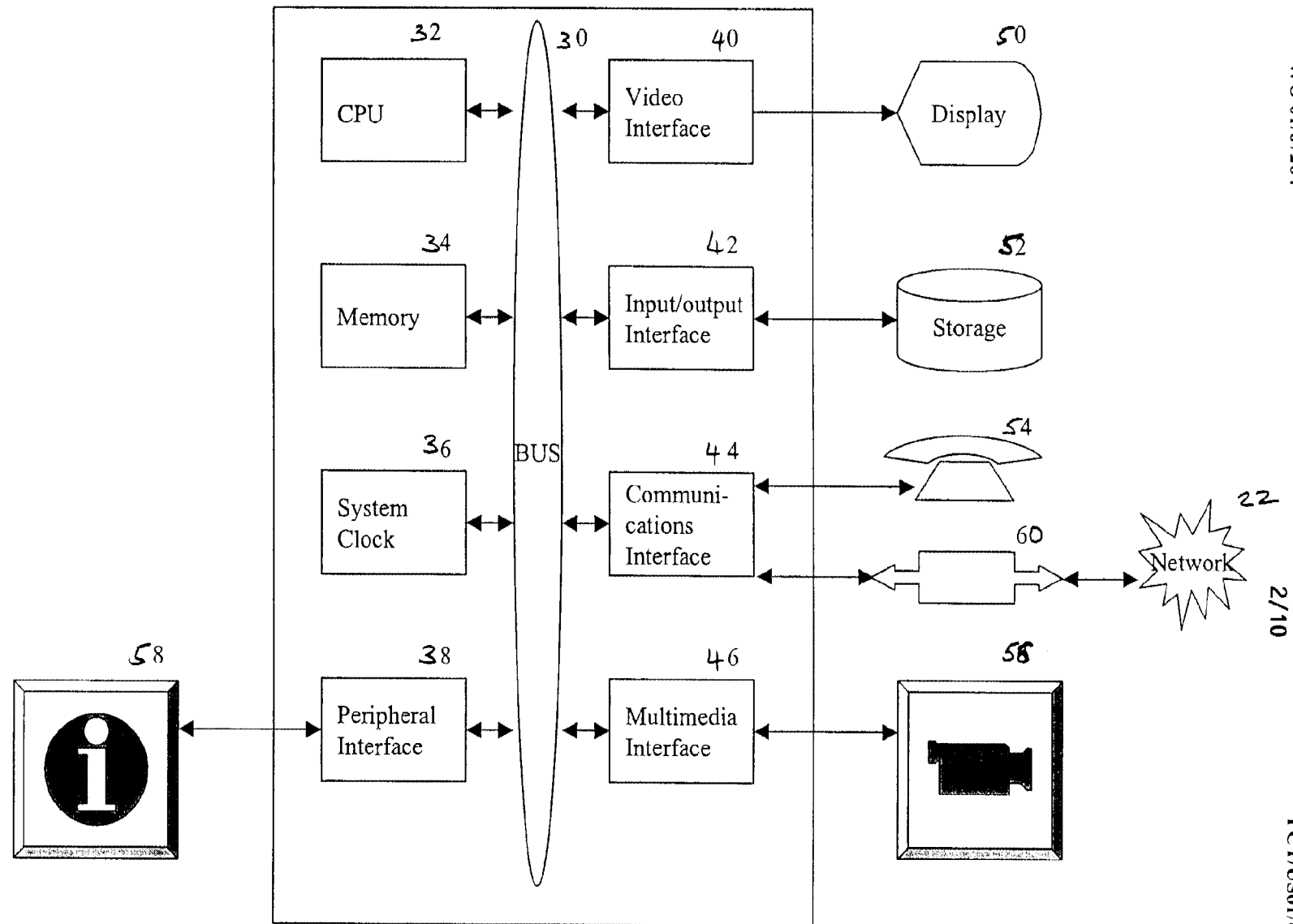


Figure 2

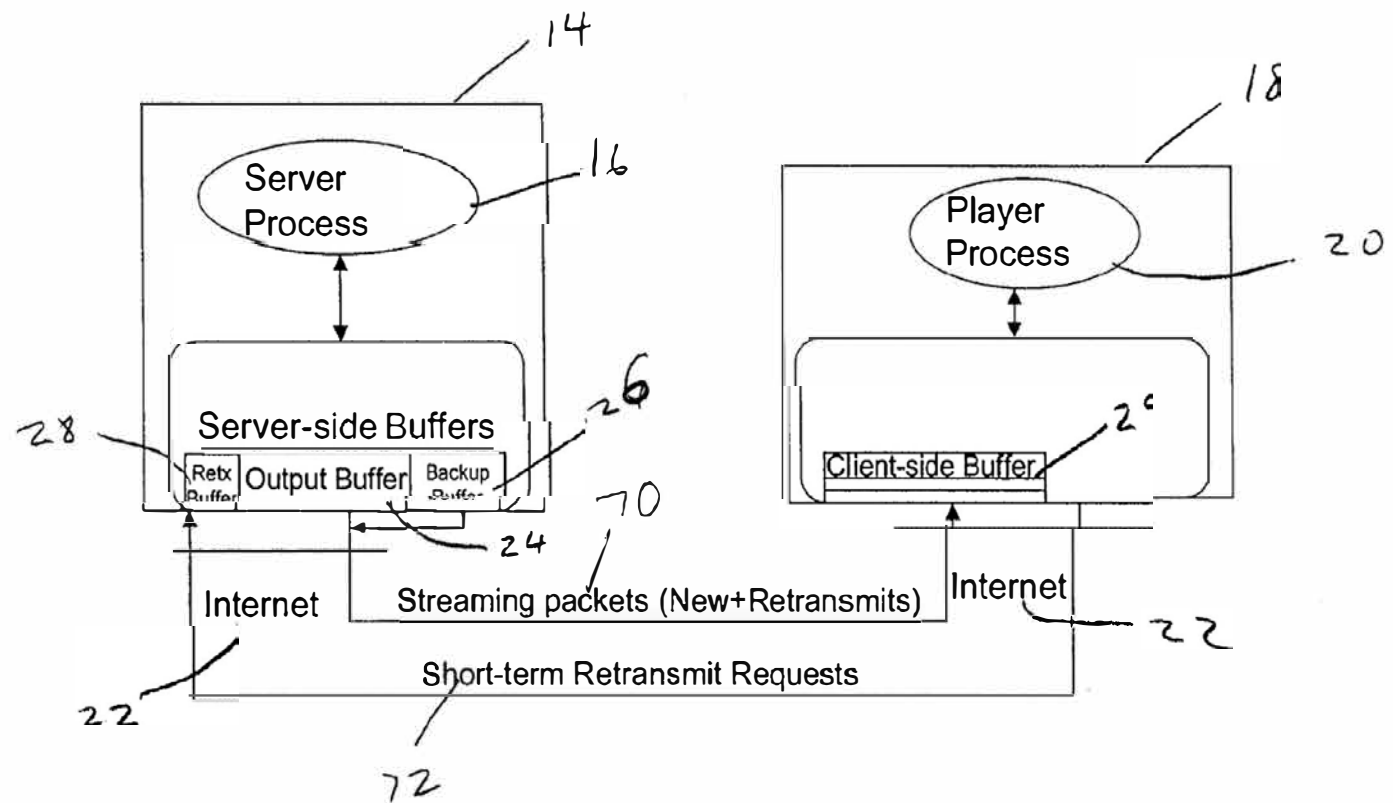


Figure 3

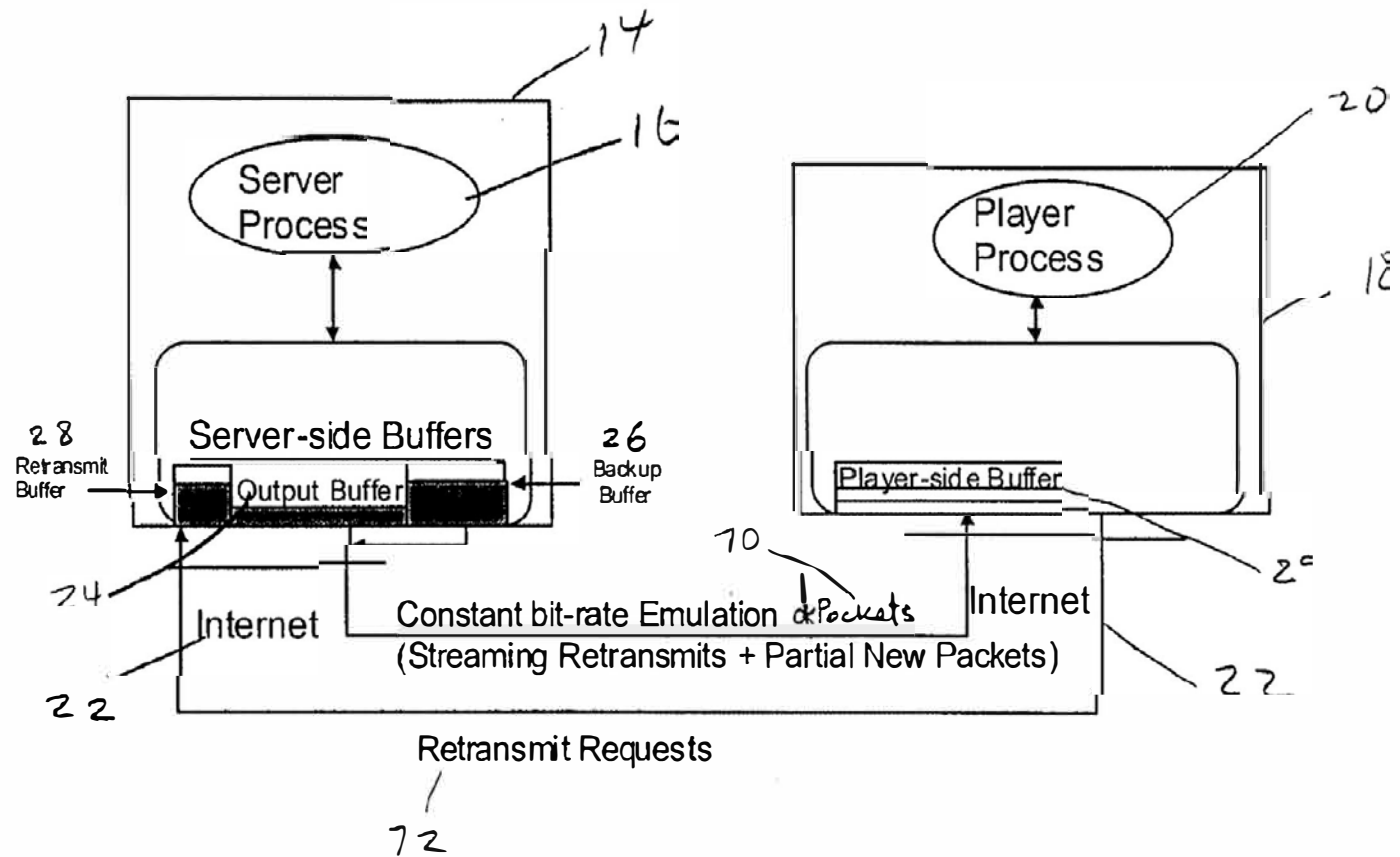


Figure 4

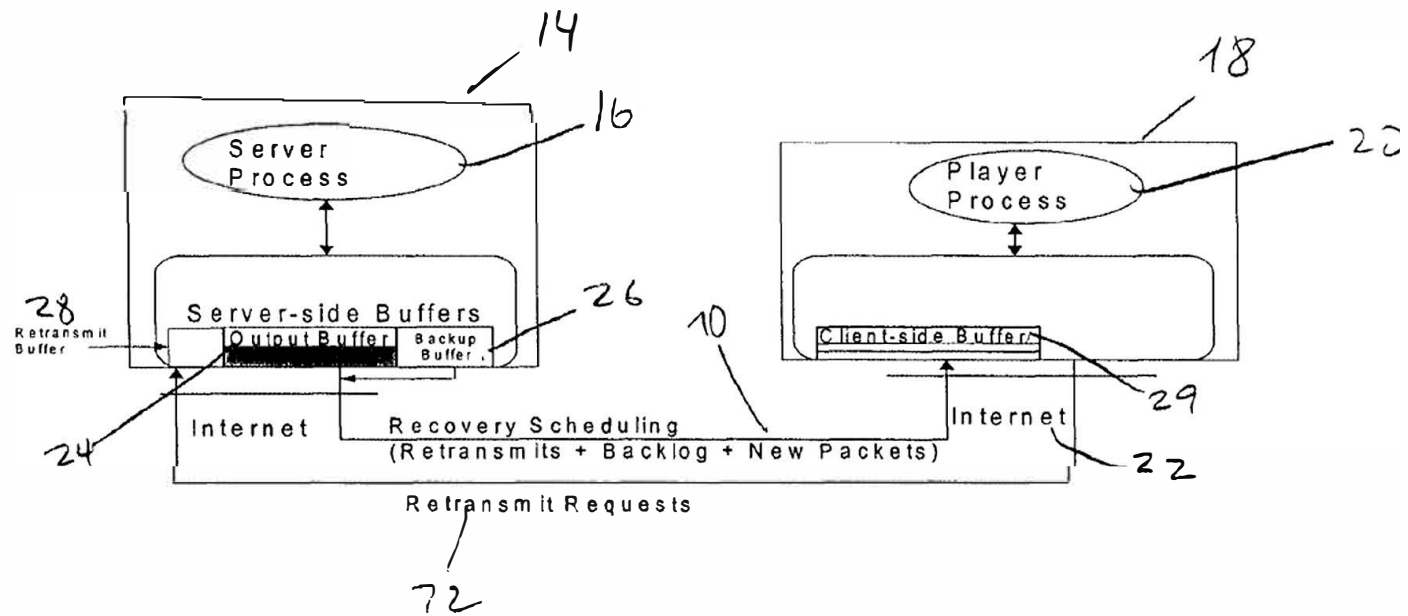


Figure 5

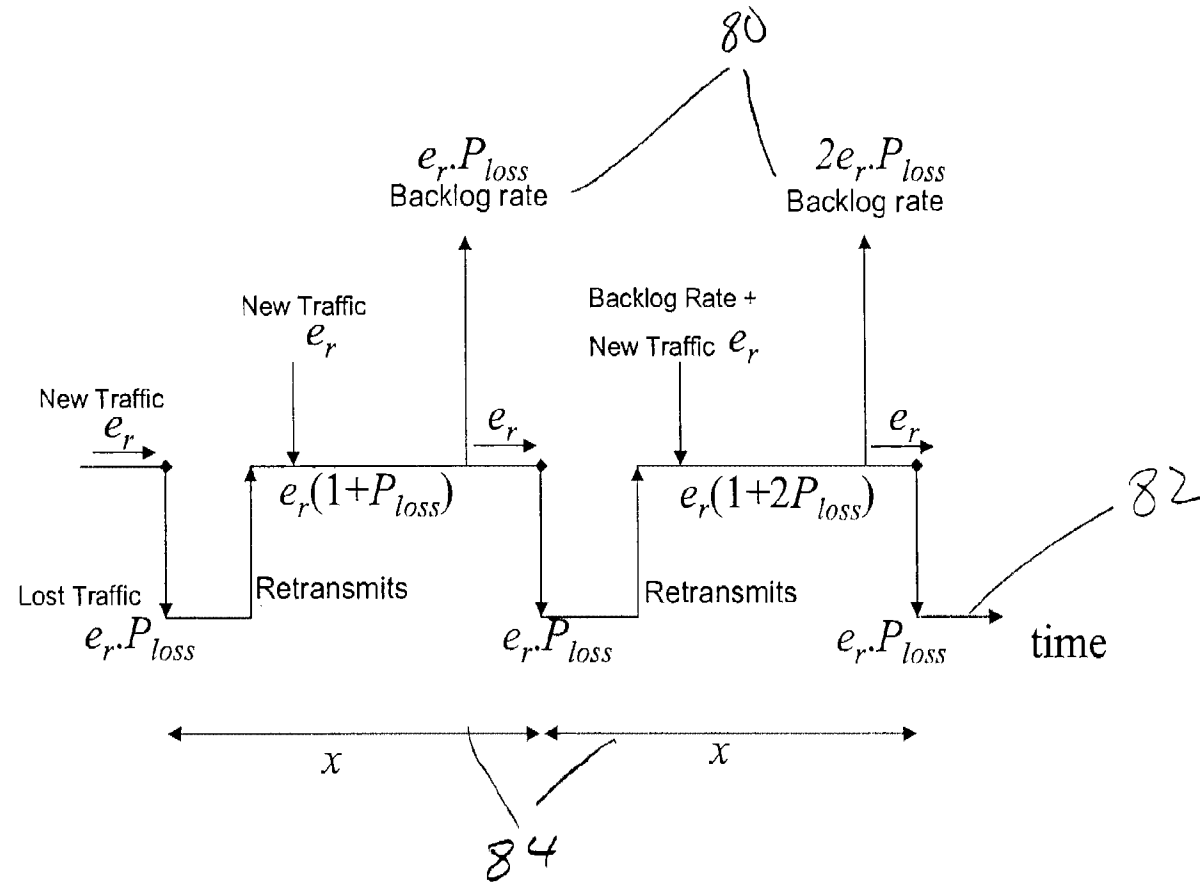


Figure 6

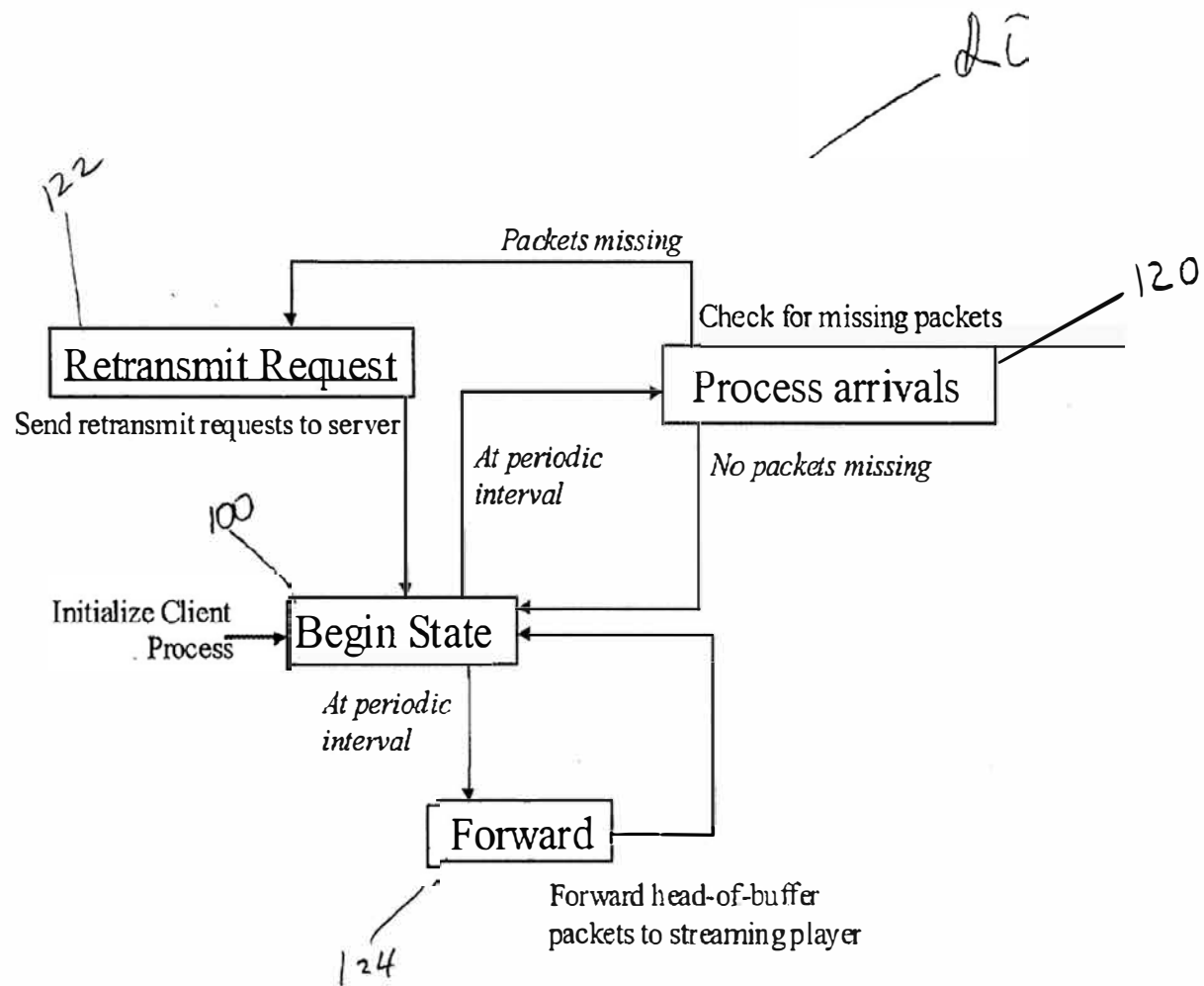


Figure 7

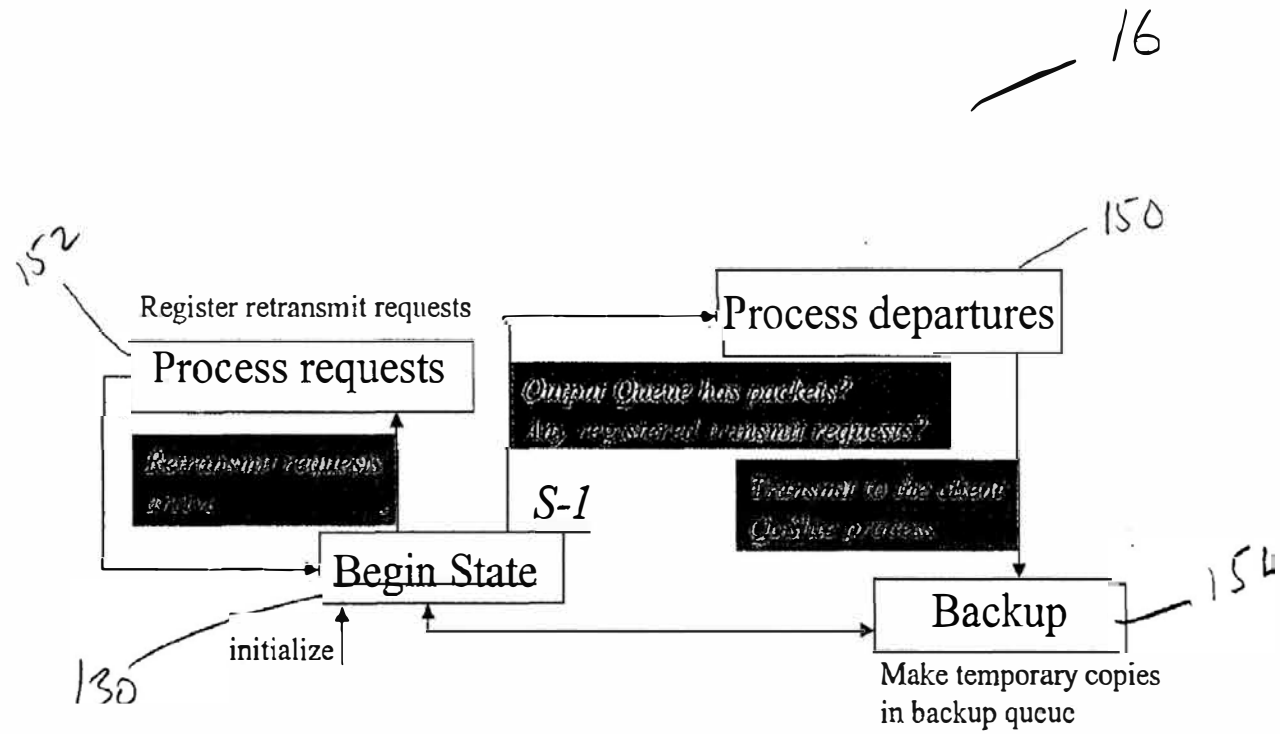


Figure 8

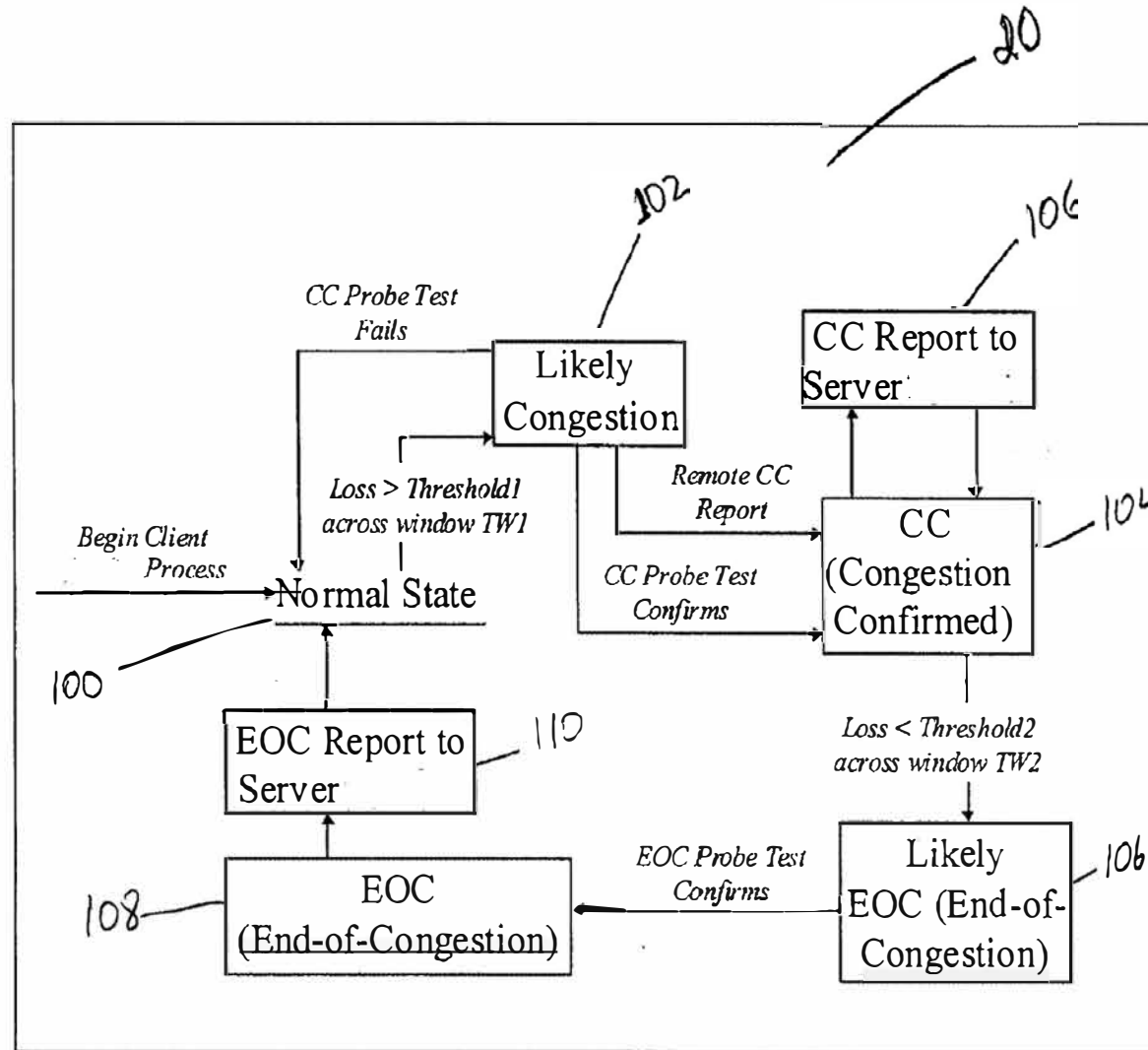


Figure 9

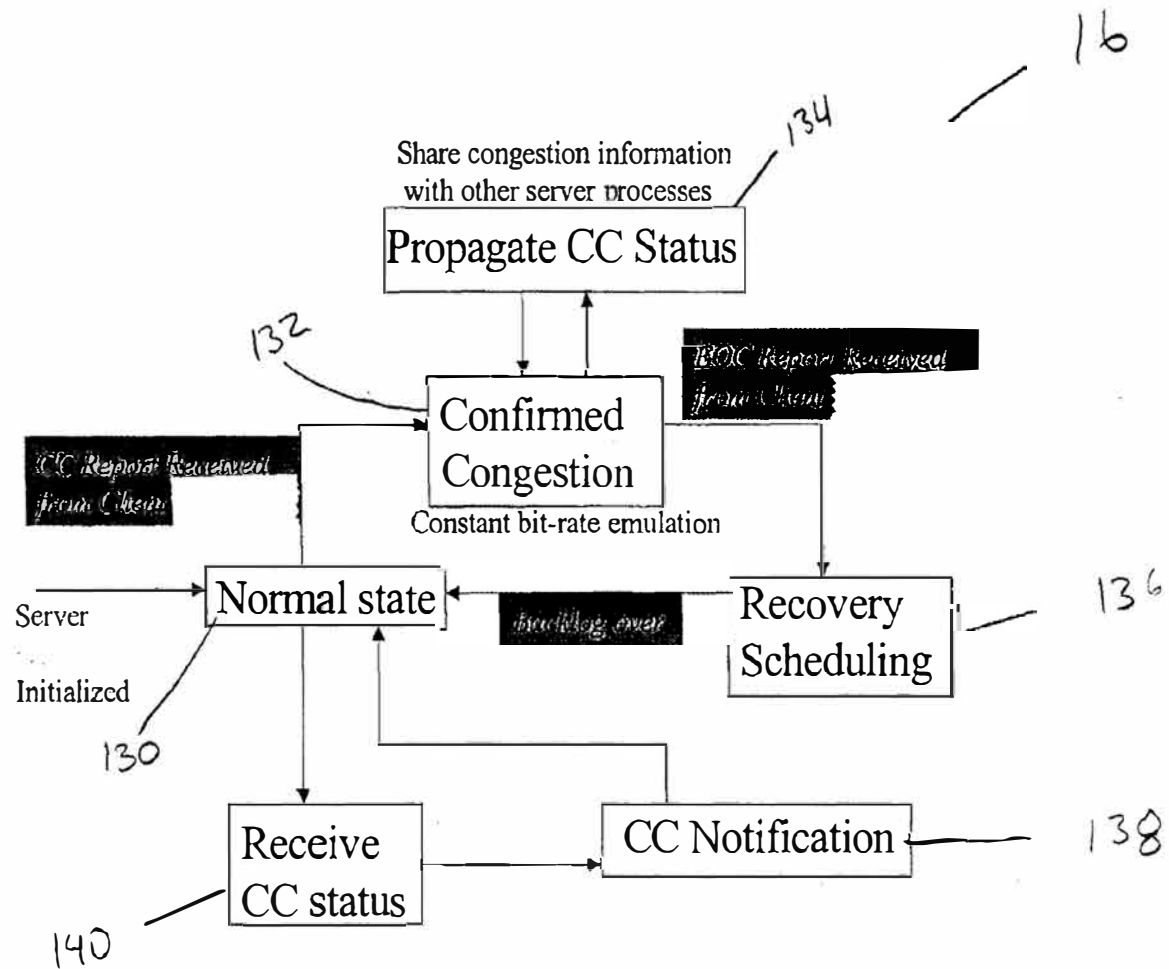


Figure 10

INTERNATIONAL SEARCH REPORT

 International application No.
PCT/US01/40264

A. CLASSIFICATION OF SUBJECT MATTER														
IPC(7) : G06F 15/16 US CL : 709/231, 232, 235 According to International Patent Classification (IPC) or to both national classification and IPC														
B. FIELDS SEARCHED														
Minimum documentation searched (classification system followed by classification symbols) U.S. : 709/231, 232, 235														
Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched														
Electronic data base consulted during the international search (name of data base and, where practicable, search terms used) EAST														
C. DOCUMENTS CONSIDERED TO BE RELEVANT														
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.												
X, P	US 6,031,818 A (LO et al) 29 February 2000, col. 2, line 54-col. 5, line 28.	1, 2												
X,P	US 6,018,5/5 A (SORBER) 25 January 2000, col. 4, line 18-col. 10, line 65.	6-10, 14-18												
A, P	US 6,014,706 A (CANNON et al) 11 January 2000, see all.	1-18												
A	US 5,339,392 A (RISBERG et al) 16 August 1994, see all.	1-18												
☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.														
<table border="0"> <tr> <td>* Special categories of cited documents:</td> <td>*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention</td> </tr> <tr> <td>*A* document defining the general state of the art which is not considered to be of particular relevance</td> <td>*X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone</td> </tr> <tr> <td>*B* earlier document published on or after the international filing date</td> <td>*Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art</td> </tr> <tr> <td>*L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)</td> <td>* & * document member of the same patent family</td> </tr> <tr> <td>*O* document referring to an oral disclosure, use, exhibition or other means</td> <td></td> </tr> <tr> <td>*P* document published prior to the international filing date but later than the priority date claimed</td> <td></td> </tr> </table>			* Special categories of cited documents:	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention	*A* document defining the general state of the art which is not considered to be of particular relevance	*X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone	*B* earlier document published on or after the international filing date	*Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art	*L* document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	* & * document member of the same patent family	*O* document referring to an oral disclosure, use, exhibition or other means		*P* document published prior to the international filing date but later than the priority date claimed	
* Special categories of cited documents:	*T* later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention													
A document defining the general state of the art which is not considered to be of particular relevance	*X* document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone													
B earlier document published on or after the international filing date	*Y* document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art													
L document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	* & * document member of the same patent family													
O document referring to an oral disclosure, use, exhibition or other means														
P document published prior to the international filing date but later than the priority date claimed														
Date of the actual completion of the international search 06 JUNE 2001		Date of mailing of the international search report 27 July 2001 (27.07.01)												
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230		Authorized officer <i>Robert Hancock</i> TOD KUPSTAS Telephone No. 703-305-2655												

Form PCT/ISA/210 (second sheet) (July 1998) *

(19) World Intellectual Property
Organization
International Bureau



(43) International Publication Date
25 March 2004 (25.03.2004)

PCT

(10) International Publication Number
WO 2004/025405 A2

(51) International Patent Classification⁷: **G06F**
(21) International Application Number:
PCT/US2003/028275
(22) International Filing Date:
10 September 2003 (10.09.2003)
(25) Filing Language: English
(26) Publication Language: English
(30) Priority Data:
60/409,645 10 September 2002 (10.09.2002) US
(71) Applicant (for all designated States except US): **THOM-
SON LICENSING S.A.** [FR/FR]; 46, Quai A. Le Gallo,
F-92648 Boulogne (FR).

(74) Agents: **TRIPOLI, Joseph, S.** et al.; Thomson Licensing
Inc., Suite #200, Two Independence Way, Princeton, NJ
08540 (US).

(81) Designated States (national): AE, AG, AL, AM, AT, AU,
AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU,
CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE,
GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR,
KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK,
MN, MW, MX, MZ, NI, NO, NZ, OM, PG, PH, PL, PT,
RO, RU, SC, SD, SE, SG, SK, SL, SY, TJ, TM, TN, TR,
TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States (regional): ARIPO patent (GH, GM,
KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),
Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE,
ES, FI, FR, GB, GR, HU, IE, IT, LU, MC, NL, PT, RO,
SE, SI, SK, TR), OAPI patent (BF, BJ, CF, CG, CI, CM,
GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

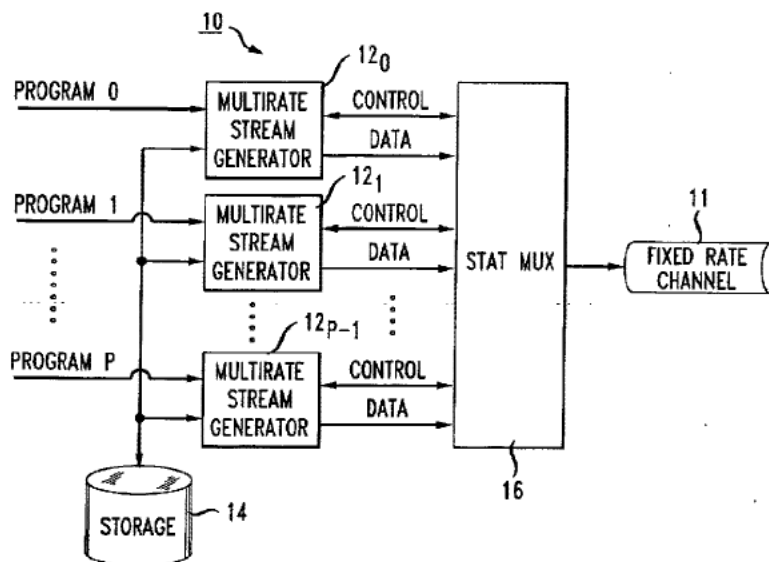
(72) Inventors; and
(75) Inventors/Applicants (for US only): **BOYCE, Jill, Mac-
Donald** [US/US]; 3 Brandywine Court, Manalapan, NJ
07726 (US). **RAMASWAMY, Kumar** [IN/US]; 71 Sayre
Drive, Princeton, NJ 08540 (US).

Published:

— without international search report and to be republished
upon receipt of that report

[Continued on next page]

(54) Title: VIDEO ON DEMAND SERVER SYSTEM AND METHOD



(57) Abstract: A Video-on-Demand Server architecture (100, 200, 300) transmits a plurality of pre-coded programs having different bit rates across a fixed bandwidth channel (11). For each program, a generator (120-12P-1, 110, 210) generates a plurality of different bit rate representations for each program. Each generator also provides control information at each of a plurality of successive time windows T for each bit rate representation. The control information provides a bit rate and a quality measure during each time window T. The control information enables a statistical multiplexer (16, 160, 260) to select a bit rate representation for each program during each time window T to maximize the quality of the selected representations while not exceeding the total available channel capacity.



For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

VIDEO ON DEMAND SERVER SYSTEM AND METHOD

TECHNICAL FIELD

5 This invention relates to a technique for efficiently providing video on demand while maintaining bandwidth constraints of a fixed channel.

BACKGROUND ART

10 Present day Video-on-Demand (VoD) systems typically transmit Constant Bit Rate (CBR) video. Transmitting CBR video does not achieve the same efficiency as transmitting variable bit rate (VBR) video for equivalent long-term average bit rates. Transmitting multiple pre-recorded VBR-encoded video programs over a constant bandwidth channel has problems since the sum of the instantaneous bitrates required by each program can sometimes exceed the
15 total available bitrate. On the other hand, broadcast applications employing real-time encoders use statistical multiplexing to exploit the instantaneous bitrate variations between multiple programs being carried on the same carrier or transponder. The encoders operate in real time to enable the system to constrain the total combined bit rate for all programs so as not to exceed the channel capacity over a given time window. Each video program has an associated complexity
20 measure. A central controller dynamically adjusts the bit rate allocated to each video program based upon the relative complexities.

 There now exist devices known as "video transraters" that operate to reduce the bit rate of video streams within a common compression standard, such as MPEG 2 for example. Present day transraters often experience difficulties when converting between constant and variable bit
25 rates. Such transraters also can experience difficulties when trying to alter the bit rate for bit streams that have many scene changes, or a large number of I-Pictures.

 Scalable video encoding, which permits dividing a video signal into a base layer and one or more enhancement layers, can also address bit rate issues. Several methods of scalable video encoding exist, including spatial, SNR, temporal, data partitioning, fine grain scalability (FGS),
30 frequency scalability. The MPEG-2 and MPEG-4 video compression standards include several scalability methods. Using scalable encoding requires both the transmitter and receiver have the same ability to implement different coding algorithms, thus introducing additional complexity.

- 2 -

Thus, there exists a need for a technique for managing variable bit rate video on demand that obviates the disadvantages of the prior art.

BRIEF SUMMARY OF THE INVENTION

5

Briefly, in accordance with present principles, there is provided a method for transmitting a plurality of pre-coded programs having different bit rates across a fixed bandwidth channel. For each program, at least two, and preferably, a plurality of different bit rate representations are generated. Control information is provided at each of a plurality of successive time windows for each bit rate representation. The control information provides a bit rate and a quality measure during each time window. The control information enables selection during each time window of a bit rate representation for each program to maximize the quality of the selected representations while not exceeding the total available channel capacity in that time window.

15 BRIEF SUMMARY OF THE DRAWING

FIGURE 1 depicts a block schematic of first preferred embodiment of a Video-on-Demand server architecture in accordance with the present principles;

20 FIGURE 2 depicts a block schematic of second preferred embodiment of a Video-on-Demand server architecture in accordance with the present principles; and

FIGURE 3 depicts a block schematic of third preferred embodiment of a Video-on-Demand server architecture in accordance with the present principles.

DETAILED DESCRIPTION

25

Before proceeding to describe the technique of the present principles, the following terminology will prove helpful:

30 T is the time interval over which the system optimization is being contemplated

C is the total channel capacity available in time frame T

P is the total number of programs

$p \in (0, P-1)$, is the index of a particular program

$N[p]$ is the total number of representations of program p

- 3 -

$n[p] \in (0, N[p] - 1)$ is the index of a particular representation of program p

$r[p, x]$ is the bit rate of representation x of program p during T

$q[p, x]$ is the quality of representation x of program p during T

5 FIGURE 1 depicts a first preferred embodiment of a Video-on-Demand server architecture 10 in accordance with the present principles for transmitting pre-encoded CBR and/or VBR audio/video programs over a fixed bandwidth channel 11 having a total channel capacity, C . The server architecture 10 includes a plurality of multirate stream generators $12_0, 12_1 \dots 12_{P-1}$, where P is an integer greater than zero corresponding to the number of separate input
10 streams. The stream rate generators 12_0-12_{P-1} each pre-code a corresponding one of the programs Program 0, Program 1...Program $P-1$ at a plurality of different bit rates. Stated another way, each multirate stream generator pre-codes the corresponding program to yield at least two or more different bit rate representations. A representation can include a particular stored encoded
15 device 14, typically in the form of a single disc drive, a Redundant Array of Inexpensive Discs (RAID), or a plurality of RAID's, serves to store the pre-coded representations generated by the multirate stream generators 12_0-12_{P-1} .

Each of the multirate stream generators 12_0-12_{P-1} generates control information for each time window T of for the corresponding one of Programs 0 to $P-1$, respectively. The control
20 information for each time window T for each representation of each program includes an indication of the bit rate of that representation and a quality measure, such as the peak signal-to-noise ratio (PSNR) of the representation. A central statistical multiplexer ("stat mux") 16 receives the control information for each representation. At each successive time window T , the stat mux 16 selects a representation for each program to maximize the quality of the transmitted
25 programs while maintaining the total bit rate at or below the total capacity C of the channel 11.

The lowest bit rate representation for each program should not exceed a prescribed value such that the sum of the lowest bit rates for all programs will not exceed the channel capacity for each interval T . In this way, at least one representation of each program can undergo transmission over the channel 11. Requiring the lowest bit rate representation for each program
30 to have a peak bit rate of C/P or less can achieve this constraint. Other methods exist that meet the constraint $\sum_{p=0}^{P-1} r[p,0] \leq C$ for all time windows T of all programs. Typically, advance knowledge does not exist as to when playback of particular programs will commence. Therefore,

- 4 -

enforcing capacity restraints becomes easiest by requiring that the bit rate limit of the lowest bit rate representation be the same for all time windows T of a particular program.

The process for the maximization of the overall combined quality can occur in several different ways. In all cases, the constraint $\sum_{p=0}^{P-1} r[p, n[p]] \leq C$ must remain met for all time

5 windows T of all programs. A minimax approach can satisfy the constraint by choosing $n[p]$ for each $p \in (0, P-1)$ to maximize the quality of the minimum quality program (i.e., minimizing the maximum distortion.)

The minimax approach can be implemented by sorting the control information for each representation of each program so that the quality and bit rate index increases monotonically increasing with the index. The change in bit rate for each step (referred to as the "delta (Δ)" bit rate) is then stored. The stat mux 16 begins with the lowest index for each program representation and computes the total capacity S. The stat mux 16 the selects the program representation at the lowest quality and checks if adding its delta bit rate to S exceeds C. If the addition of the representation does not exceed C, the index for that representation is incremented, and process is repeated. Once C is exceeded, the representation with the next to lowest quality is checked to see if adding its delta bit rate to S exceeds C. The process is repeated until no increment in any representation can be made without exceeding the channel capacity C. Alternatively, the total quality of all programs could be optimized by maximizing the sum of the individual program qualities which involves solving the following constrained optimization

20 problem:

$$\max_{n[\cdot]} \sum_{p=0}^{P-1} q[p, n[p]]; \text{ subject to } \sum_{p=0}^{P-1} r[p, n[p]] \leq C$$

It is also possible to optimize the product of the individual program qualities in the following manner:

$$\max_{n[\cdot]} \prod_{p=0}^{P-1} q[p, n[p]]; \text{ subject to } \sum_{p=0}^{P-1} r[p, n[p]] \leq C$$

25 A weighted average can also be used, in order to provide different classes of service for different viewers.

The multirate stream generators 120-12_{P-1} can use several different methods, or a combination of methods to form multiple representations of each program. In all cases, a random access point, such as an intra-coded (I) frame should occur at the beginning of each time window T, for each representation, so that for each time window T a different representation can be

30

- 5 -

chosen without causing drift. Each representation can use either CBR or VBR coding as long as the capacity of the lowest bit rate representation meets the total channel capacity constraint.

In a first method, a multirate video encoder encodes each program at several different bit rates into several independent bit streams. Each different bit rate bit stream serves as a different representation. FIGURE 2 depicts an architecture 100 for carrying out this method. As seen in FIG. 2, a multirate video encoder 110 serves to encode a corresponding program to yield a plurality of different rate representations. A separate one of transport packetizers 112₀, 112₂...112_{P-1} each packetizes a corresponding representation. The multirate encoding performed by the multirate encoder 110, and the packetization performed by the packetizers 112₀-112_{P-1} occurs once, not necessarily in real time, prior to storage in a storage device 140 similar in construction to the storage device 14 of FIG. 1.

Selection of the representations stored in storage device 140 for output in real time occurs in response to a signal received from a selector block 150. The selector block 150 identifies the stored representation for output in response to a request from the stat mux 160 for a particular representation. Upon receipt of the signal from the selector block 150, the storage device 140 supplies the selected packetized representations to the stat mux 160 for output.

In a second method, portions of the several different bit streams are combined to yield additional representations of a program. The presence of random access points at the same location in all bit streams of a program (as would naturally occur at scene changes) or the use of fixed group-of-pictures structures, permits the combination of compressed data from different bit rate streams at random access boundaries to form new representations. No need exists to store each representation independently, as long as the ability exists to generate each representation from the data that is stored. Consider the following example where T is one second, the data is coded at 30 fps, and I frames are inserted every 15 frames, yielding three different bit rate bit streams, designated as Bit stream 0, Bit stream 1, and Bit Stream 2, respectively. Each bit stream constitutes a separate representation. An additional representation can be formed which uses Bitstream 0 for the first 15 frames and Bitstream 1 for the second 15 frames, and so on.

Assuming alignment of the random access points with the transport packets, the server architecture 100 of FIG. 2 works well when the transport packetization occurs in advance.

FIGURE 3 depicts a VoD server architecture 200 better suited when the random access points do not necessarily align with the transport packets. Referring to FIG. 3, VoD server architecture 200 includes multirate video encoder 210 for generating a plurality of different bit rate

- 6 -

representations for storage in a storage device 240 similar to the storage devices 14 and 140. The representations stored in the storage device 140 exist as un-packetized bit streams.

Selection of the representations stored in storage device 240 for output in real time occurs in response to a signal received from a selector block 250. The selector block 250, similar in nature to the selector block 150 of FIG. 2, identifies the stored representation for output in response to a request from a static mixer/multiplexer 260 for a particular representation. Upon receipt of the signal from the selector block 250, the storage device 240 supplies the selected packetized representations to the static mixer/multiplexer. A packetizer 212 packetizes the output stream of the static mixer/multiplexer 260 for output on the channel (not shown).

Representations also can be formed by switching between stored bit streams for non-reference pictures, such as for video codecs (not shown) that include pictures that are not used as reference pictures for predicting other pictures, such as MPEG-2 B frames or JVT non-stored pictures. Each non-reference picture can be selected from a different stored bitstream without affecting the quality of subsequently coded pictures, as the non-stored pictures are not used for prediction. Complete elimination of a non-reference picture in a representation can also occur. Each non-reference picture could be switched individually, or groups of non-reference pictures could be switched together, allowing for many possible total bit rate representations. The multirate stream signal generators 12_0-12_{P-1} of FIG. 1 can choose to limit the number of possible representations to send to the stat mux 16 of FIG. 1 to choose those with significant differences in quality and bit rate. It is not necessary for each representation to be independently stored in storage, but a table listing the location and length of each coded frame can be stored to simplify generation of the representation when needed. It is also possible to store multiple bit rate bit streams only for the non-stored pictures and only store a single version of the reference (I and P) pictures, which can significantly reduce storage requirements.

With the VoD server architectures 100 and 200 of FIGS. 2 and 3, respectively, the stat mux 160 and the static mixer/multiplexer 260 generate a bit stream corresponding to the selected representation. If individual pictures are not transport packet aligned, the architecture 200 of FIG. accomplishes transport packetization after the generation of the bit stream for a particular representation. Otherwise, the VoD server architecture 100 of FIG. 2 will work well.

In accordance with another aspect of the present principles, scalable video coding can occur by the use of a scalable video encoder (not shown) to form a base layer and one or more enhancement layers. Any type of scalability that which employs the base layer for motion compensated reference picture prediction in order to avoid drift can be used, such as frequency

- 7 -

scalability, FGS, SNR scalability or temporal scalability. The lowest bit rate representation corresponds to the base layer. The peak bit rate of the base layer must be such that when the bit rates of all programs' base layers are summed, the channel capacity C is not exceeded for each time window T . Other higher bit rate representations will correspond to the base layer plus some portion of the enhancement layer. This approach requires that the video decoder support the type of scalability used. The VoD server architecture 100 of FIG. 2 is used upon inclusion of an entire enhancement layer in a particular representation. The VoD server architecture 200 of FIG. 3 is used if portions of an enhancement layer are used in a representation.

For systems which use encryption for contents right management, the VoD server architecture 100 of FIG. 2 does not require any decryption or encryption in real time, as encryption is added to the transport packetization process done in non-real time and entire encrypted transport packets are stored and transmitted. The VoD server architecture 200 of FIG. 3 requires that encryption be performed along with transport packetization in real-time.

VoD programs generally include both audio and video. As audio bit rates are generally significantly lower than video bit rates and are also generally at a fixed bit rate, there is less advantage to be gained by selecting from different pre-encoded audio bit streams by the stat mux 16 and 160 of FIGS 1 and 2, and the static mixer/multiplexer 260 of FIG. 3. A user likely will find the switching of audio quality during the viewing of a single program disturbing. The total channel capacity C can be considered to be totally consumed by the video channel; assuming that a single bit rate audio stream is also sent for each program. Alternatively, the above-described methods can also be applied to audio, where multiple representations of audio programs with different bit rates and qualities can be generated.

A video player (not shown) that works with the VoD server architectures of the present principles will contain a video decoder (not shown) and some storage for buffering. For the particular program being received, the data can arrive at a non-uniform data rate, but with the requirement that the data corresponding to time unit T will arrive any time within the T time window. The video player must have the capability of buffering and delaying for T . An upper limit on the maximum bit rate that can be transmitted for a given representation of a program for a time window T can be pre-determined to limit the decoder complexity and decoder buffer size. The decoding buffer is generally a requirement of a video decoder, e.g. MPEG-2 levels sets buffer size particular requirements, and the combination of T and the bitrate may be chosen to meet it.

- 8 -

Fast forward and fast reverse tracks for each program can also be stored at the VoD server. They can be stored at a single bit rate, or at several different bit rates, and allowed to adapt as is done for the normal play program.

- 5 Determination of the time window T length is based on several factors. T should be as large as possible to get the most statistical multiplexing gain, but small enough to constrain the delay at the video player for start-up and switching between normal play and trick play streams. T should be small enough to meet decoder requirements.

The foregoing describes a system and method for efficiently providing video on demand while maintaining bandwidth constraints of a fixed channel.

- 9 -

CLAIMS

1 1. A method for transmitting a plurality of pre-coded programs having different bit
 2 rates across a fixed bandwidth channel, comprising the steps of:
 3 generating at least two different bit rate representations of each program;
 4 providing control information at each of a plurality of successive time windows T for
 5 each representation of each program, the control information for each successive window
 6 indicating a bit rate and quality measure for a representation of a corresponding program; and
 7 during each time window T, selecting a representation for each program such to
 8 maximize the quality of the selected representations while not exceeding a total available
 9 capacity for the channel.

1 2. The method according to claim 1 wherein the step of generating at least two
 2 different bit rates representation further comprises the step of generating for each program a
 3 lowest bit rate representation having a peak bit rate not greater than C/P where C is the total
 4 channel capacity in time T and P is the total number of programs.

1 3. The method according to claim 1 wherein the step of providing the control
 2 information further comprises the step of establishing the peak signal-to-noise ratio (PSNR) as
 3 the quality measure embodied in the control information.

1 4. The method according to claim 1 wherein the selecting step further comprises the
 2 step of selecting a representation for each program which meets the constraint

$$3 \quad \sum_{p=0}^{P-1} r[p, n[p]] \leq C \text{ for all time windows wherein:}$$

4 C is the total channel capacity available in time frame T;

5 P is the total number of programs;

6 $p \in (0, P-1)$, is the index of a particular program;

7 $N[p]$ is the total number of representations of program p ;

8 $n[p] \in (0, N[p] - 1)$ is the index of a particular representation of program p ; and

9 $r[p, x]$ is the bit rate of representation x of program p during T

- 10 -

5. The method according to claim 4 further comprising the step of choosing each program's representation $n[p] \in (0, N[p]-1)$ to maximize the quality of the program p that had the minimum quality.

6. The method according to claim 5 further comprising the steps of:

- (a) sorting the quality information for with the bit rate and quality measure monotonically increasing with an index value;
- (b) storing each bit rate increment (delta) and quality value for each index value;
- (c) beginning with a lowest index value, computing total capacity S for program representations selected thus far for such index value;
- (d) selecting a program representation at a lowest quality measure;
- (e) checking whether the bit rate increment of the selected program at the lowest quality, when added to the representations selected thus far, exceeds total channel capacity, and if not
- (f) incrementing the index value; and
- (g) repeating steps (c)-(f).

7. The method according to claim 1 wherein the selecting step further comprises the step of selecting the representation for each program such to maximize a sum of individual

program qualities by solving $\max_{n[p]} \sum_{p=0}^{P-1} q[p, n[p]]$; subject to $\sum_{p=0}^{P-1} r[p, n[p]] \leq C$

wherein ,

C is the total channel capacity available in time frame T ;

P is the total number of programs;

$p \in (0, P-1)$, is the index of a particular program;

$N[p]$ is the total number of representations of program p ;

$n[p] \in (0, N[p] - 1)$ is the index of a particular representation of program p ;

$r[p, x]$ is the bit rate of representation x of program p during T ; and

$q[p, x]$ is the quality of representation x of program p during T .

8. The method according to claim 1 wherein the selecting step further comprises the step of selecting the representation for each program such to maximize a product of individual program qualities by solving

- 11 -

$$\max_{n[p]} \prod_{p=0}^{P-1} q[p, n[p]]; \text{ subject to } \sum_{p=0}^{P-1} r[p, n[p]] \leq C$$

- 5 where,
 6 C is the total channel capacity available in time frame T ;
 7 P is the total number of programs;
 8 $p \in (0, P-1)$, is the index of a particular program;
 9 $N[p]$ is the total number of representations of program p ;
 10 $n[p] \in (0, N[p] - 1)$ is the index of a particular representation of program p ;
 11 $r[p, x]$ is the bit rate of representation x of program p during T ; and
 12 $q[p, x]$ is the quality of representation x of program p during T .

1 9. The method according to claim 8 further comprising the step of applying a
 2 weighted average to provide different classes of service for different viewers.

1 10. A system for transmitting a plurality of pre-coded programs having different bit
 2 rates across a fixed bandwidth channel, comprising the steps of:
 3 means for generating at least two different bit rate representations of each program;
 4 means providing control information at each of a plurality of successive time windows T
 5 for each representation of each program, the control information for each successive window
 6 indicating a bit rate and quality measure for a representation of a corresponding program; and
 7 means for selecting during each time window T a representation for each program such to
 8 maximize the quality of the selected representations while not exceeding a total available
 9 capacity for the channel.

1 11. The system according to claim 10 wherein the generating means and control
 2 information providing means collectively comprise:
 3 a plurality of multirate stream generators, each associated with a corresponding one of the
 4 plurality of pre-coded programs.

1 12. The system according to claim 10 wherein the generating means and control
 2 information providing means collectively comprise:
 3 a multirate video encoder for encoding at least two bit rate representations of each pre-
 4 coded program.

- 12 -

1 13. The system according to claim 10 wherein the generating means and control
 2 information providing means collectively comprise:
 3 a multirate video encoder for encoding at least two bit rate representations of each pre-
 4 coded program; and
 5 a plurality of transport packetizers, each serving to packetize the bit rate presentations for
 6 each pre-coded program.

1 14. The system according to claim 10 wherein the selecting means includes a static
 2 multiplexer.

1 15. The system according to claim 12 wherein the selecting means comprises:
 2 a static multiplexer; and
 3 a transport packetizer for packetizing the selecting representation.

1 16. The system according to claim 10 wherein the selecting means generates for each
 2 program a lowest bit rate representation having a peak bit rate not greater than C/P where C is the
 3 total channel capacity in time T and P is the total number of programs.

1 17. The system according to claim 10 wherein control information providing means
 2 establishes quality measure in accordance with a peak signal-to-noise ratio (PSNR).

1 18. The system according to claim 10 wherein the selecting means selects a
 2 representation for each program which meets the constraint $\sum_{p=0}^{P-1} r[p, n[p]] \leq C$ for all time

3 windows where:

4 C is the total channel capacity available in time frame T ;

5 P is the total number of programs;

6 $p \in (0, P-1)$, is the index of a particular program;

7 $N[p]$ is the total number of representations of program p ;

8 $n[p] \in (0, N[p] - 1)$ is the index of a particular representation of program p ; and

9 $r[p, x]$ is the bit rate of representation x of program p during T .

- 13 -

19. The system according to claim 18 wherein the selecting means chooses each program's representation $n[p] \in (0, N[p]-1)$ to maximize the quality of the program p that had the minimum quality.

20. The system according to claim 10 wherein the selecting means selects the representation for each program such to maximize a sum of individual program qualities by solving:

$$\max_{n[p]} \sum_{p=0}^{P-1} q[p, n[p]]; \text{ subject to } \sum_{p=0}^{P-1} r[p, n[p]] \leq C$$

where,

C is the total channel capacity available in time frame T ;

P is the total number of programs;

$p \in (0, P-1)$, is the index of a particular program;

$N[p]$ is the total number of representations of program p ;

$n[p] \in (0, N[p] - 1)$ is the index of a particular representation of program p ;

$r[p, x]$ is the bit rate of representation x of program p during T ; and

$q[p, x]$ is the quality of representation x of program p during T .

21. The system according to claim 10 wherein the selecting means selects the representation for each program such to maximize a product of individual program qualities by solving:

$$\max_{n[p]} \prod_{p=0}^{P-1} q[p, n[p]]; \text{ subject to } \sum_{p=0}^{P-1} r[p, n[p]] \leq C$$

where,

C is the total channel capacity available in time frame T ;

P is the total number of programs;

$p \in (0, P-1)$, is the index of a particular program;

$N[p]$ is the total number of representations of program p ;

$n[p] \in (0, N[p] - 1)$ is the index of a particular representation of program p ;

$r[p, x]$ is the bit rate of representation x of program p during T ; and

$q[p, x]$ is the quality of representation x of program p during T .

- 14 -

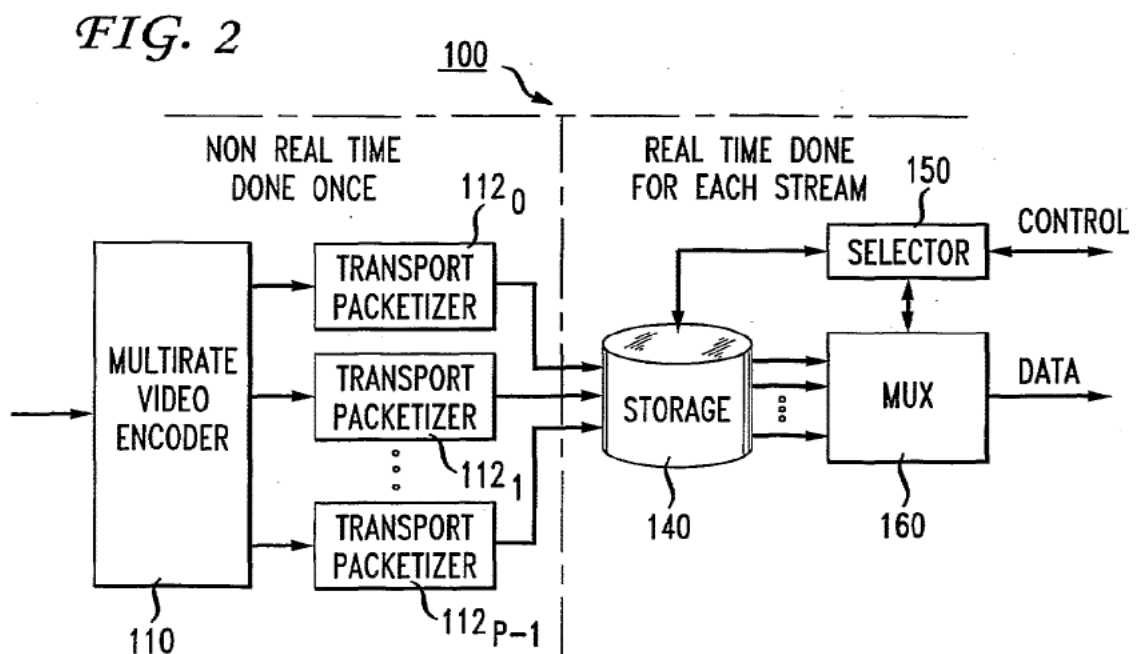
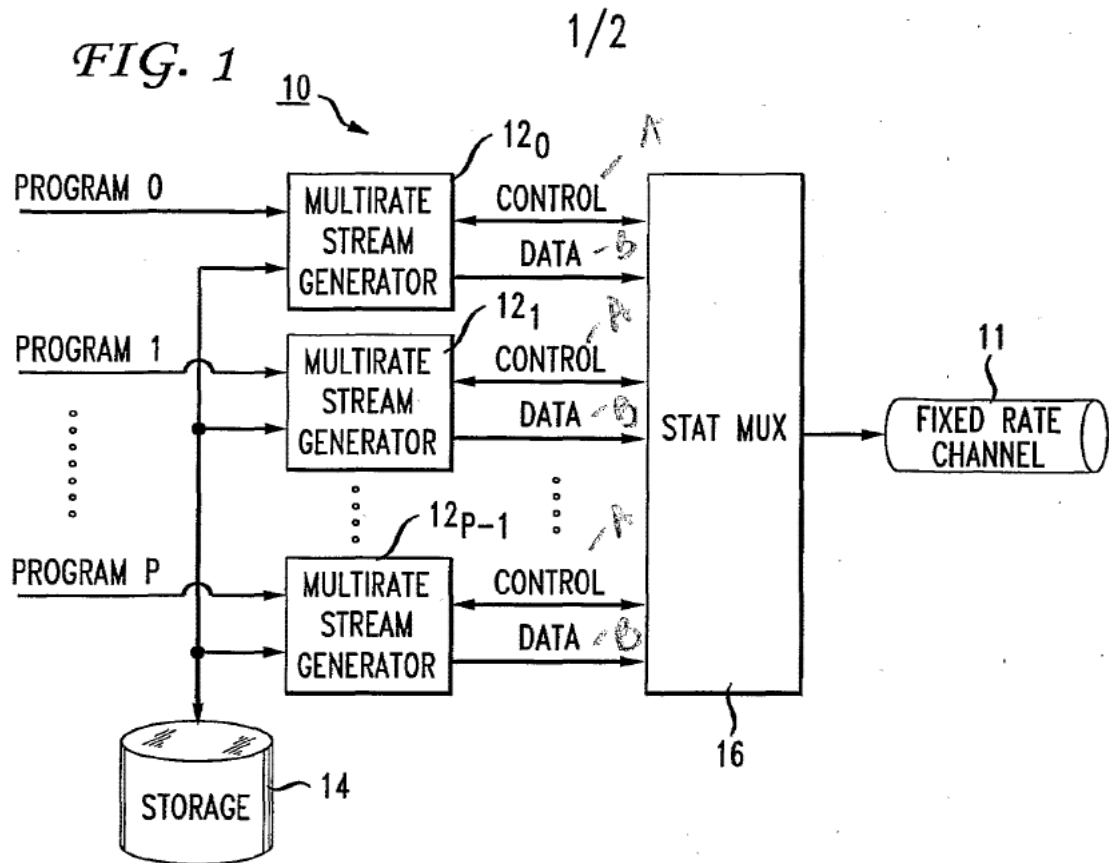
1 22. The system according to claim 10 wherein a weighted average is applied to
2 provide different classes of service for different viewers.

1

- 15 -

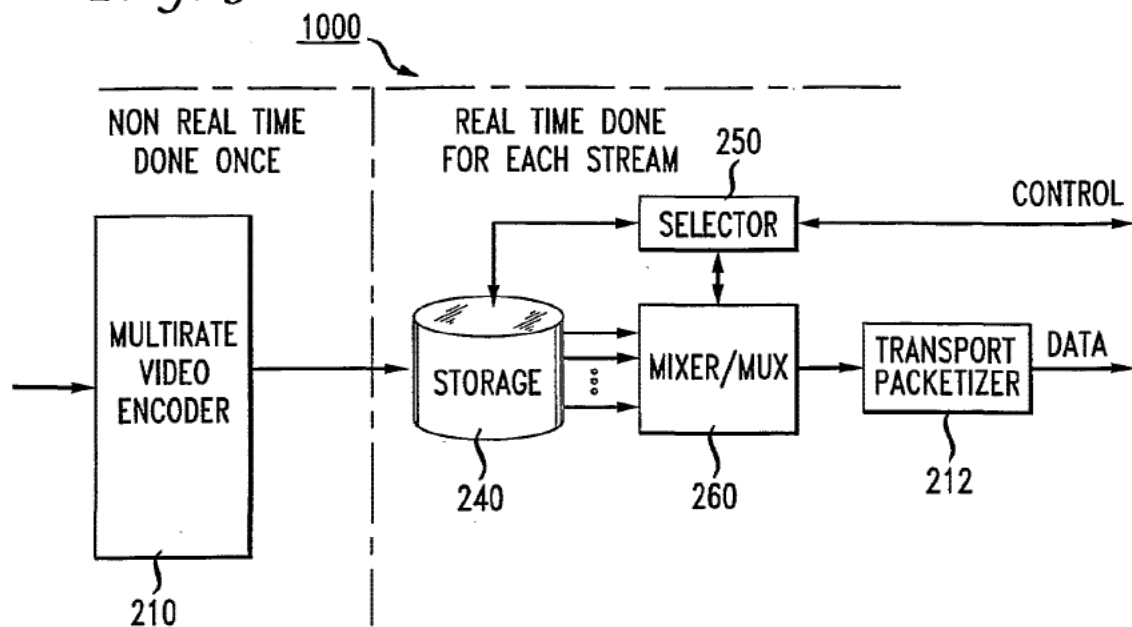
ABSTRACT OF THE DISCLOSURE

A Video-on-Demand Server architecture (100, 200, 300) transmits a plurality of pre-coded programs having different bit rates across a fixed bandwidth channel (11). For each
5 program, a generator (12₀-12_{P-1}, 110, 210) generates a plurality of different bit rate representations for each program. Each generator also provides control information at each of a plurality of successive time windows T for each bit rate representation. The control information provides a bit rate and a quality measure during each time window T. The control information enables a statistical multiplexer (16, 160, 260) to select a bit rate representation for each program
10 during each time window T to maximize the quality of the selected representations while not exceeding the total available channel capacity.



2/2

FIG. 3



PCT New International Application Inventory of Unscannable or Missing Items	Serial Number
--	---------------

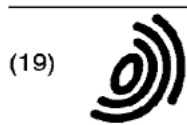
Check This Column if Item Is Present	Item	Check This Column if Item Is Missing on Filing
☒	Return Receipt Postcard	
	Check (amount \$ _____)	☒
☒	PCT EASY Diskette	
	DNA Diskette	
	Exhibit	
☒	Express Mail Label or Envelope	
	Applicant Supplied Priority Document	
	Other (specify)	XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX XXXXXXXXXXXXXXXX	Cover Letter	
	Other (specify)	

Winword® & Automation/Unscannable inventory sheet

Electronic Patent Application Fee Transmittal

Application Number:	11116783			
Filing Date:	28-Apr-2005			
Title of Invention:	Apparatus, system, and method for adaptive-rate shifting of streaming content			
First Named Inventor/Applicant Name:	R. Drew Major			
Filer:	Brett Carlson			
Attorney Docket Number:	066.0141 (P2004-04-01.1)			
Filed as Large Entity				
Utility under 35 USC 111(a) Filing Fees				
Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Basic Filing:				
Pages:				
Claims:				
Miscellaneous-Filing:				
Petition:				
Patent-Appeals-and-Interference:				
Post-Allowance-and-Post-Issuance:				
Extension-of-Time:				

Description	Fee Code	Quantity	Amount	Sub-Total in USD(\$)
Miscellaneous:				
Submission- Information Disclosure Stmt	1806	1	180	180
Total in USD (\$)				180



Europäisches Patentamt
European Patent Office
Office européen des brevets



(11) **EP 1 395 014 A1**

(12) **EUROPEAN PATENT APPLICATION**

(43) Date of publication:
03.03.2004 Bulletin 2004/10

(51) Int Cl.7: **H04L 29/06, H04L 29/08**

(21) Application number: **02019052.6**

(22) Date of filing: **27.08.2002**

(84) Designated Contracting States:
**AT BE BG CH CY CZ DE DK EE ES FI FR GB GR
IE IT LI LU MC NL PT SE SK TR**
Designated Extension States:
AL LT LV MK RO SI

• **Burmeister, Carsten**
22081 Hamburg (DE)
• **Hakenberg, Rolf**
64295 Darmstadt (DE)

(71) Applicant: **MATSUSHITA ELECTRIC INDUSTRIAL
CO., LTD.**
Kadoma-city, Osaka 571-8501 (JP)

(74) Representative: **Grünecker, Kinkeldey,
Stockmair & Schwanhäusser Anwaltssozietät**
Maximilianstrasse 58
80538 München (DE)

(72) Inventors:
• **Rey, José Luis**
64287 Darmstadt (DE)

(54) **A method of transmitting data streams with data segments of variable length**

(57) A method of transmitting data streams between a sending host and a receiving host using a transport protocol comprising the steps of making available data streams at the sending host in the form of mono-rate or multi-rate content streams having the same content but

different encoding rates, establishing a connection between the sending host and the receiving host, transmitting data streams with a selected encoded rate to the receiving host, and separating the data streams into data segments of variable length at the sending host as required during transmission.

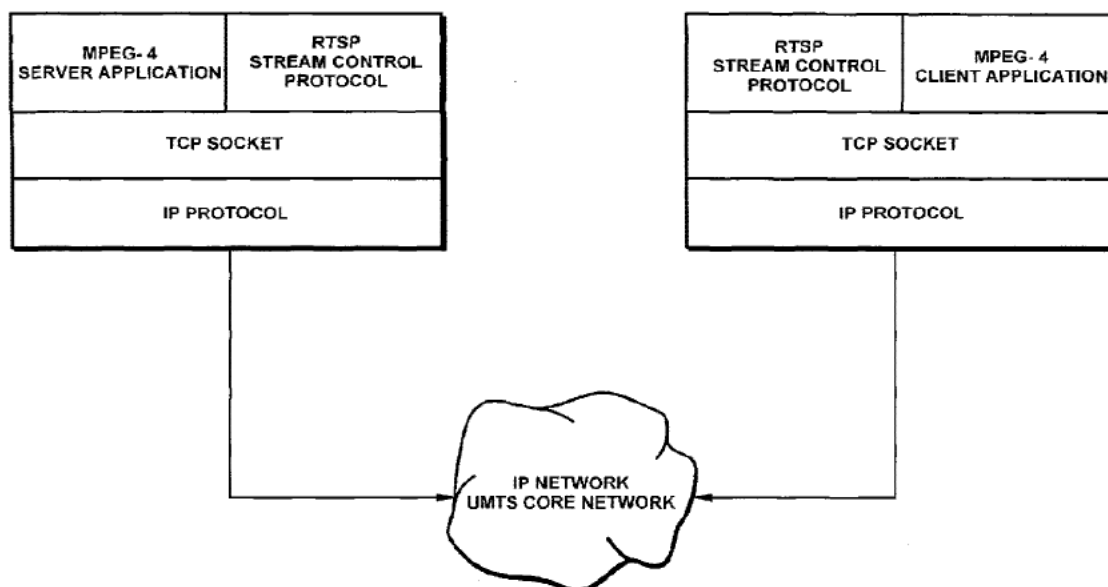


Fig. 1

Description

[0001] The present invention relates to a method of transmitting data streams between a sending host and a receiving host using a transport protocol such as TCP.

[0002] In network communications between a sending host and a receiving host, the transmission of data packets is mostly done on a best-effort basis. In the communication link, the available bandwidth and the current link conditions vary and hence, the end hosts must be provided with mechanisms to probe and adjust themselves to the current link conditions.

[0003] With the development of 3rd generation networks, so-called streaming applications are gaining importance. Streaming applications are understood as a continuous stream of data, e.g. video or audio data which is typically sensitive to time delays. In order that the user will be able to receive multimedia data streams on a portable receiver, for example a mobile phone, several requirements on the transport mechanism will be imposed in order that the delivered data meets with the resulting time constraints. These constraints are difficult to meet in a best-effort environment, which is subject to bandwidth variations and delays.

[0004] To cope with the problem of bandwidth adaptation, it is common praxis to make several data streams with different encoding rates but same content available at the sending host. According to the available bandwidth, it is then switched between the data streams in order to adjust the rate to the available bandwidth. In other words, multi-rate content streams at the sending host are transmitted and switching operations between streams are performed depending on the available bandwidth. The current stream is forwarded to the receiving host in one piece and consequently, for every switching operation, the ongoing connection must be torn down and a new one must be set up. As a result, this operation is inefficient and decreases the data throughput.

[0005] Alternatively, it is possible to transmit only one stream of data to the receiving host and to discard data packets based on priority and time constraints. However, this leads to a loss of data and is therefore not an optimal solution.

[0006] The object subject to the present invention is to minimize the latency of the switching operation and to ensure fast recovery from an incorrect switching decision.

[0007] This object is solved according to the present invention with a method as set forth in claim 1. Preferred embodiments of the method are subject to various dependent claims.

[0008] The method of the present invention supposes the existence of multi-rate content data streams at the sending host. With the method of the present invention, it is possible to quickly switch between data streams that are variably segmented at the application layer as required during transmission. Instead of using data

streams of fixed segment length, the data segments given to the socket are made smaller before, during and after the switching. For each content several video streams encoded with different rates are stored at the video server. Under normal sending conditions, i.e. the transmission has reached some equilibrium where the bandwidth is enough and the losses are not dominating, the video content is passed to the sending TCP socket in segments of the same size (irregardless of the stream encoding rate, the segment size is a function of the link, not of the encoding rate). Only when the server is notified (or knows by other means) that a stream switch might be needed, then it reduces the size of these segments.

[0009] Further, with the method of the present invention, it is neither necessary to modify the originally encoded data at the sending host or to modify the transport protocol nor to introduce additional control messages, making the method easy to implement.

[0010] Preferably, the switching is performed between data streams of different rate and the variable segmentation of the data streams is performed around the switching instant. Hence, a quick stream switching from the old to the new data stream can be performed.

Further, preferably, the segmentation during, before or after the switching instance is made finer, such that the switching operation is more resilient to an incorrect decision and enables a quicker, further switching operation.

[0011] Advantageously, the step of switching between streams involves a feedback message (e.g. an RTSP command) i.e. communication between hosts when a switching operation is needed and when to make the necessary preparations therefor, for example, to start with finer segmentation.

[0012] It is furthermore of advantage if the feedback message includes at least one of the parameters, which comprise the length of data segment, an identifier of the data to be sent or a stream control command which makes the operation smoother and less susceptible to data losses.

[0013] Preferably, the length of the data segments is gradually increased after a switching instant to reduce interface communication, i.e. between application and TCP socket.

[0014] The present invention will be more fully understood from the following detailed description with reference to the accompanying drawings which show:

Figure 1: the basic architecture of a sending host and a receiving host connected by a communications network in which the method subject to the present invention can be performed,

Figure 2: a block diagram of a server architecture as the sending host in which the method of the present invention is performed, and

Figure 3: a block diagram of a client architecture.

[0015] Figure 1 illustrates the basic architecture of a communications network for connecting a video server as a sending host to a video client acting as a receiving host. The communications network is, for example, a UMTS core network, an external network, such as the internet or an internal intranet over which the server and the client communicate by means of a transport protocol. As transport protocol TCP is selected as an example, but it is clear to a skilled person that any other transport protocol for transmitting data can be employed.

[0016] In a more general way, it will now be explained how TCP handles the exchange of data between the server and the client.

[0017] First of all, the video server needs to transfer its data to the protocol and tell it to send same to the client. To progressively read and send data, the protocol uses a kind of buffer with added functionalities, a so-called TCP socket. Generally, sockets are unequivocally identified by the IP addresses of the correspondents, the protocols and ports which are used. Once a connection between two sockets is established, the communication can be controlled by use of interface calls, which, for example may comprise "create a socket", "bind a socket to a transport address", "accept" or "close a connection", "send", "receive" or "write" data between the sender and the receiver.

[0018] Wide-spread socket implementations are UNIX BSD 4.2 and Windows Winsock 2.0, where equivalent interface calls are implemented.

[0019] Each interface call has parameters, e.g. for the "send" call, parameters of relevance are the socket descriptor, the identifier of the data to be sent and the length of the data stream. Once a data stream is passed in one call, no single bit of this data stream can be modified. The only way to abort transfer of this data stream is to break down the connection.

[0020] Nevertheless, in order to close the connection, the application must explicitly issue a "close" call. Interface calls can be queued as, for example, in FTP. When asked for a file, the FTP server usually issues two interface calls for this transmission, namely a "send" call indicating the socket, the identifier of the data stream and the length thereof and a "close" call identifying the socket. The two interface calls are put into the queue and when the "send" call is completed, the connection is irreversibly broken down because the "close" call was next in the queue.

[0021] Alternatively, for example, in case of TELNET, the connection is established and the socket is set to wait for incoming data, i.e. commands typed on the keyboard.

[0022] Of course, the application can be configured such that the server only issues a send call and waits for the client to close. However, this is uncommon.

[0023] Additionally, it is assumed that the TCP protocol provides a means for asynchronously signaling cer-

tain information about events in the socket. Often in the specification, the information will be an error message. In other cases, there will be information relating to the completion of processing a SEND or RECEIVE or another interface call.

[0024] Referring again to Figure 1, the video server provides, e.g. an MPEG 4 compressed data stream, i.e. a given video content, which is made available to the TCP socket in the form of mono-rate or multi-rate streams. Multi-rate streams are defined for the purpose of this and the following description, as one or several data streams with the same video content at different encoding rates. The MPEG data streams are forwarded to the TCP socket. Subsequently, by means of an IP protocol, data is transmitted to the video client. A stream control protocol, e.g. RTSP, should be used to perform such operating as PLAY, PAUSE or STOP the transmission.

[0025] With reference to Figure 2, which shows the basic architecture of a video server 200, at the MPEG server application 210, video content is made available in a buffer 211 at different coding rates. A switch 212, which receives stream control RTSP commands from RTSP block 220, such as "Play", "Pause", "Tear down" or "Change rate", selects one of the data streams which is forwarded to the TCP socket 230 in form of an MPEG file.

[0026] It is clear to a skilled person that other stream control protocols than RTSP could be used at the server and the client. The stream control protocol typically instructs to perform play, forwarding, rewinding, pause or stop functions. Additionally, if multi-rate content is available, switch to a higher or lower encoding rate can be instructed.

[0027] Once a server has received a request for a file containing a chosen content encoded at a desired rate, the server application forwards each of the segments, in which the file was previously segmented, one after another to the TCP socket by means of consecutive interface calls, e.g. consecutive SEND calls. Unlike in current FTP applications, the connection is not torn down after each data transmission process is completed. The data connection remains open, so that there is no need to build up a new connection every time a segment is transmitted. Following this, the TCP socket forms with these segments packet data units (PDUs) for transmission to the client, as it finds most appropriate

[0028] Upon establishing a connection, the TCP socket selects a segment size for the data streams which the server finds most appropriate for the link under consideration of the channel bandwidth and starts transmitting.

[0029] Initially, the size of the segments is made small before, during and after a switching operation in order to avoid a large waiting time before a further switch operation can be performed in case the previous switch decision was incorrect. Subsequently, the size of the data segments can be increased gradually throughout the connection time.

[0030] Eventually, when there is a change in the available link bandwidth, the video client shall indicate a possible switch operation by means of, for example, an RTSP message and the server will start doing finer segmentation. Hence, the server switches between data streams of different encoding rates. This is done based on initial information and subsequently controlled by feedback from the client. Alternatively, this could be determined by the server, which would, however, impose processing requirements on the server. Also a combination of control mechanisms at the server and the client is possible. How the server or the client exactly makes the decision to perform a switching operation or indicates a possible anticipated switch is out of the scope of the present invention.

[0031] The basic architecture of a TCP streaming client 300 is illustrated in Figure 3. The client 300 typically consists of a TCP socket 310, an application layer buffer 320 and a buffer manager 330 for the application layer buffer.

[0032] The client 300 processes the data segments of variable length as they come. In response thereto, it sends ACKs and performs flow control in order to prevent an overflow of the TCP socket 310 while extracting the maximum bit rate possible from the link. To do this, the TCP socket indicates to the server, with each ACK, how much data it can receive until the socket is full using a window field in the header of the TCP protocol. If needed, the TCP client signals a zero-window telling the server to enter a "Persist" mode wherein no data is sent until the client allows to do so while the connection is maintained. The Persist mode and the signaling of a window field are standard functionalities of the TCP protocol. When the TCP circuit has capacity for more data, it signals this to the server. Streaming control commands are signalled to the server by means of RTSP commands.

[0033] The application layer buffer 320, also called "play-out buffer" receives data from the socket and transmits same to the MPEG client application 340. The manager 330 for the application layer buffer 320 receives a frame request from the MPEG client application and issues a socket request upon considering the state of the application layer buffer. The buffer manager 320 therefore insures that the bit rate with which the TCP socket 310 can deliver the new data to the application layer buffer 320 matches the bit rate at which the application layer buffer is emptied to the MPEG client application 340. The exact control mechanism of the buffer manager is out of the scope of the present invention.

[0034] Finally, from the MPEG client application, the data is transferred to a display 350 for visualization for the user.

Claims

1. A method of transmitting data streams between a sending host and a receiving host using a transport

protocol comprising the steps of:

making available data streams at the sending host in form of mono-rate or multi-rate content streams having same content but different encoding rates,

establishing a connection between the sending host and the receiving host for the duration of transmission,

transmitting data streams of the same content with a selected encoding rate to the receiving host, and

separating the data streams in data segments of variable length at the sending host as required during the established connection.

2. The method according to claim 1, further comprising the steps of switching between data streams of different rate and performing the variable segmentation of the data streams around the switching instant.

3. The method according to claim 1 or 2, wherein the data streams are finer segmented during, before or after the switching instant.

4. The method according to claim 2, wherein the step of switching between streams involves a feedback message.

5. The method according to claim 4, wherein the feedback message includes signalling at least one of the parameters which comprise data segment length, an identifier of the data to be sent or a stream control command.

6. The method according to one of claims 1-5, wherein the receiving host communicates to the sending host when a stream switching is needed.

7. The method according to one of claims 1-6, wherein the sending host determines the timing of the switching instant dependent on the available bandwidth of the communication link.

8. The method according to one of claims 1-7, wherein the length of the data segments is gradually increased after a switching instant.

9. The method according to one of claims 1-8, wherein the received data streams at the receiving host are temporarily stored in an internal buffer.

10. The method according to one of claims 1-9, wherein the receiving host issues stream control commands

to the sending host depending on the state of the internal buffer.

5

10

15

20

25

30

35

40

45

50

55

5

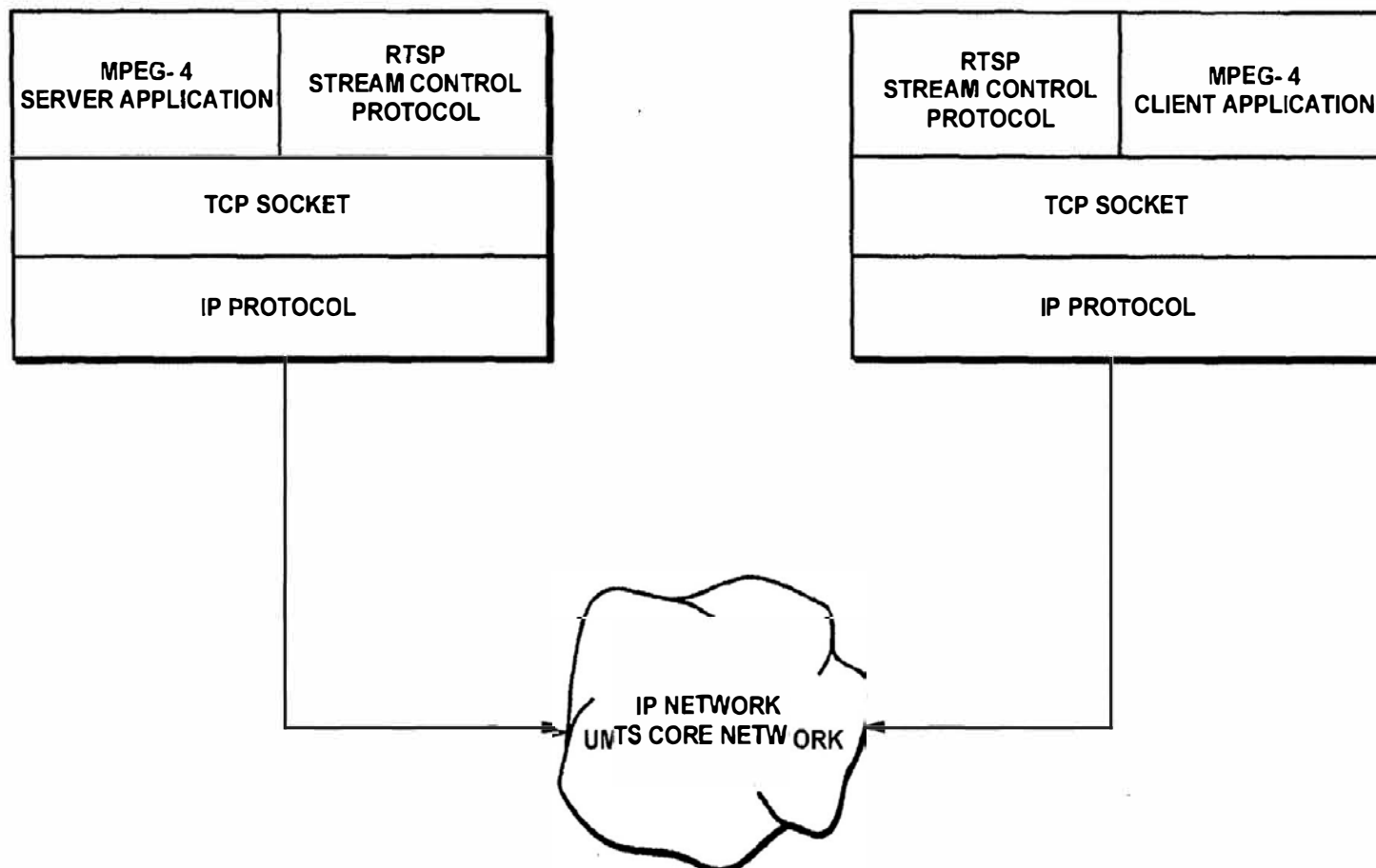


Fig. 1

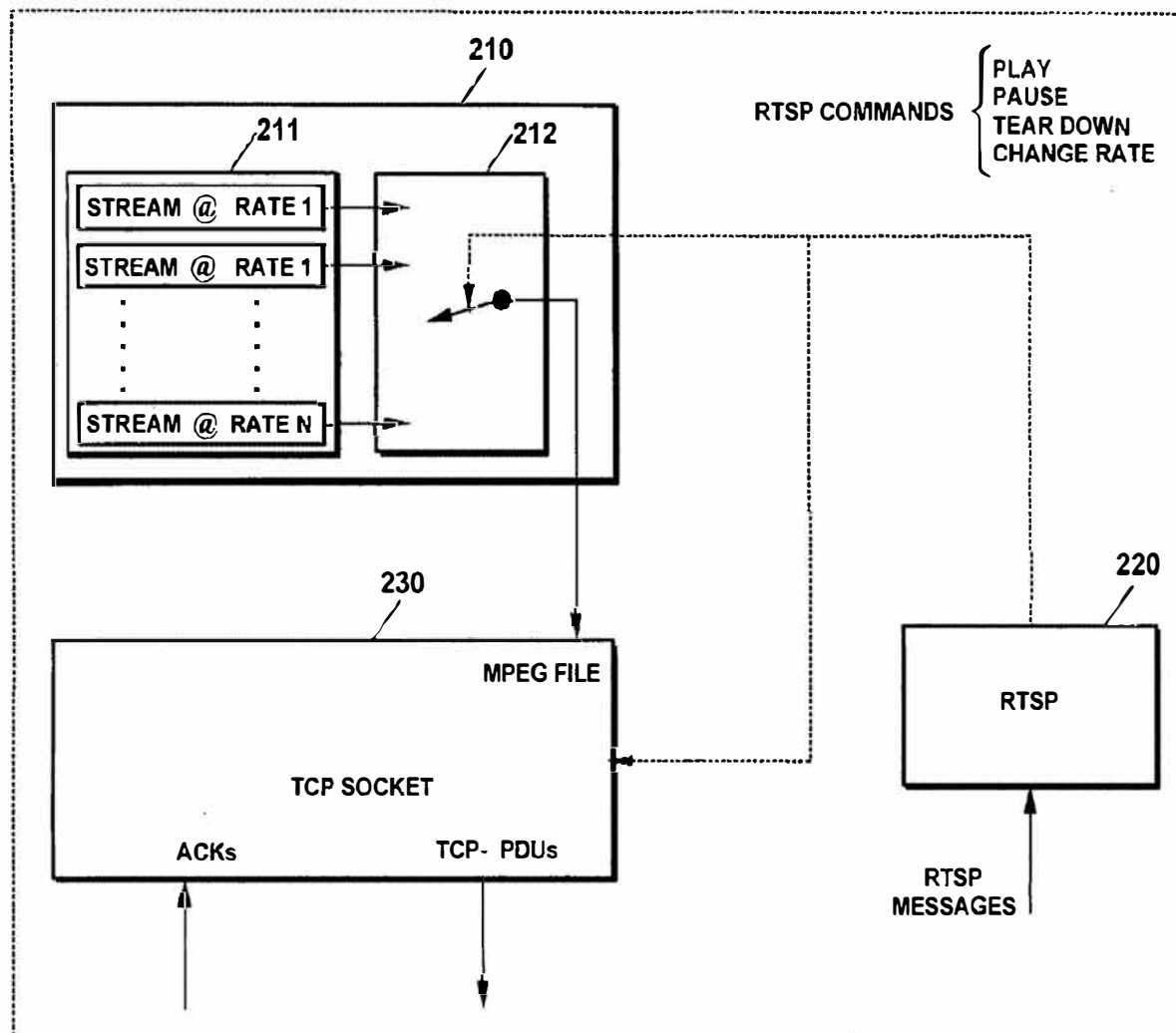


Fig. 2

200

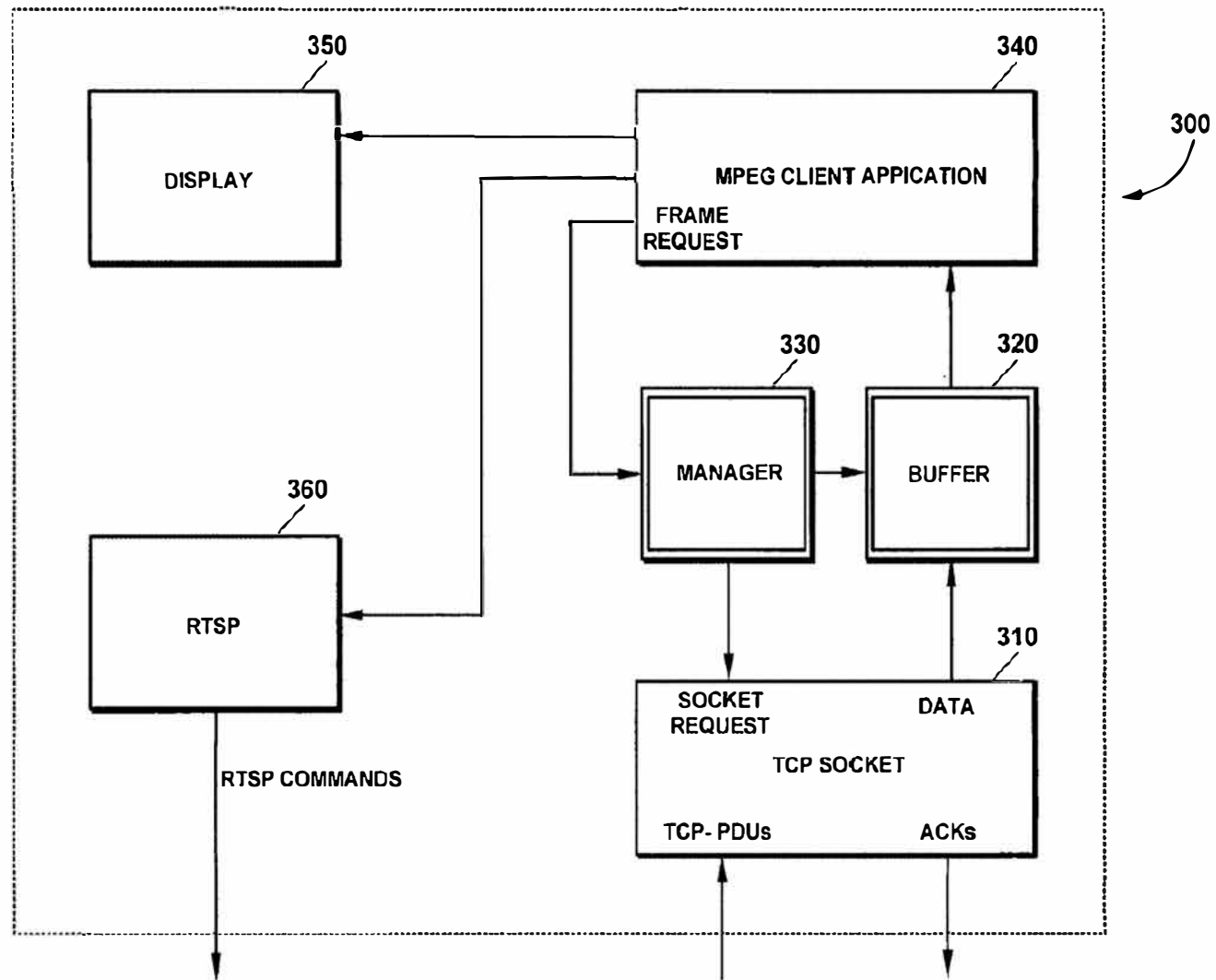


Fig. 3



European Patent
Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 01 9052

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (Int.Cl.7)
Y	CONKLIN G J ET AL: "VIDEO CODING FOR STREAMING MEDIA DELIVERY ON THE INTERNET" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, IEEE INC. NEW YORK, US, vol. 11, no. 3, March 2001 (2001-03), pages 269-281, XP001093477 ISSN: 1051-8215 * page 271, left-hand column, line 35 - line 44; figures 1,2 * * page 270, left-hand column, line 11 - line 20 * ---	1-10	H04L29/06 H04L29/08
Y	EP 0 786 885 A (NOKIA TELECOMMUNICATIONS OY) 30 July 1997 (1997-07-30) * abstract * * page 2, column 2, line 14 - line 34 * ---	1-10	
X	WO 02 054284 A (LEISTAD GEIRR I ;DANKWORTH HARALD (NO); FAST SEARCH & TRANSFER ASA) 11 July 2002 (2002-07-11) * page 2, line 31 - page 3, line 7 * * page 4, line 6 - page 5, line 6; figure 1 * * page 9, line 18 - page 10, line 15 * * page 11, line 3 - line 28; figure 3 * --- -/--	1	
			TECHNICAL FIELDS SEARCHED (Int.Cl.7) H04L
The present search report has been drawn up for all claims			
Place of search MUNICH		Date of completion of the search 7 January 2003	Examiner Michael, T
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document			

EPO FORM 1503 (03.02) (P04C01)



European Patent
Office

EUROPEAN SEARCH REPORT

Application Number
EP 02 01 9052

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION, (Int.Cl.7)
A	HUI J Y ET AL: "CLIENT-SERVER SYNCHRONIZATION AND BUFFERING FOR VARIABLE RATE MULTIMEDIA RETRIEVALS" IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS, IEEE INC. NEW YORK, US, vol. 14, no. 1, 1996, pages 226-237, XP000548824 ISSN: 0733-8716 * page 228, left-hand column, line 15 - line 33 * * page 228, right-hand column, line 14 - line 32 *	10	TECHNICAL FIELDS SEARCHED (Int.Cl.7)
A	JONGWON KIM ET AL: "TCP-friendly Internet video streaming employing variable frame rate encoding and interpolation" IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, vol. 10, no. 7, October 2000 (2000-10), pages 1164-1177, XP002226167 ISSN: 1051-8215 * page 1164, right-hand column, line 25 - line 40; figure 1 *	7	
A	WO 02 052859 A (DURRANT SIMON ; LAKE JULIAN (GB); MULROY PAT (GB); PA CONSULTING SE) 4 July 2002 (2002-07-04) * page 3, line 12 - line 30 *	7	
The present search report has been drawn up for all claims			
Place of search MUNICH		Date of completion of the search 7 January 2003	Examiner Michael, T
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document			

EPO FORM 1523 03.82 (PUB/C01)

**ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.**

EP 02 01 9052

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

07-01-2003

Patent document cited in search report		Publication date		Patent family member(s)	Publication date
EP 0786885	A	30-07-1997	FI	960382 A	27-07-1997
			EP	0786885 A2	30-07-1997
			US	5914960 A	22-06-1999

WO 02054284	A	11-07-2002	NO	20010056 A	05-07-2002
			WO	02054284 A1	11-07-2002

WO 02052859	A	04-07-2002	WO	02052859 A1	04-07-2002

EPO FORM P0458

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2011-004225

(43)Date of publication of application : 06.01.2011

(51)Int.Cl.

H04W 48/20 (2009.01)

H04W 84/12 (2009.01)

H04W 48/16 (2009.01)

(21)Application number : 2009-146288

(71)Applicant : NEC INFRONTIA CORP

(22)Date of filing : 19.06.2009

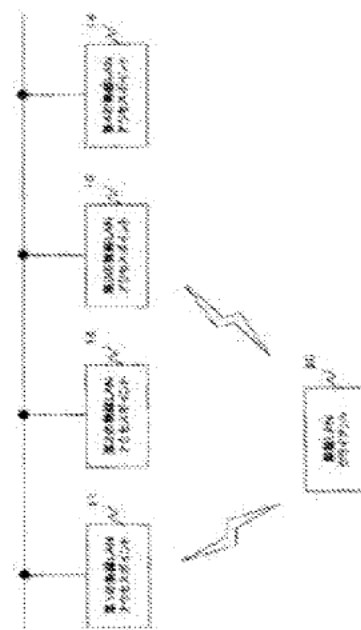
(72)Inventor : HODOSHIMA TAKESHI

(54) SYSTEM, METHOD AND PROGRAM OF HIGH-SPEED ROAMING OF WIRELESS LAN

(57)Abstract:

PROBLEM TO BE SOLVED: To attain high-speed roaming in a wireless LAN network.

SOLUTION: In a LAN system including: a plurality of wireless LAN access points; and a wireless LAN client to be connected to the wireless LAN access points, the wireless LAN access points are the ones existing on the same local area network, used channels and BSSIDS of the wireless LAN access points set to the same ESSID are recognized as wireless LAN access point information, held as data, and the wireless LAN client utilizes the wireless LAN access point information when the above connection is performed.



(19) 日本国特許庁(JP)

(12) 公開特許公報(A)

(11) 特許出願公開番号

特開2011-4225

(P2011-4225A)

(43) 公開日 平成23年1月6日(2011.1.6)

(51) Int. Cl.	F I	テーマコード (参考)
H04W 48/20 (2009.01)	H04Q 7/00 416	5K067
H04W 84/12 (2009.01)	H04Q 7/00 630	
H04W 48/16 (2009.01)	H04Q 7/00 404	

審査請求 未請求 請求項の数 15 O L (全 11 頁)

(21) 出願番号	特願2009-146288 (P2009-146288)	(71) 出願人	000227205
(22) 出願日	平成21年6月19日 (2009.6.19)		N E C インフロンティア株式会社
			神奈川県川崎市高津区北見方2丁目6番1号
		(74) 代理人	100130029
			弁理士 永井 道雄
		(74) 代理人	100166338
			弁理士 関口 正夫
		(74) 代理人	100152054
			弁理士 仲野 孝雅
		(72) 発明者	程島 健
			神奈川県川崎市高津区北見方2丁目6番1号
			N E C インフロンティア株式会社内
		Fターム (参考)	5K067 AA15 EE02 EE10 HH22 HH23 JJ72

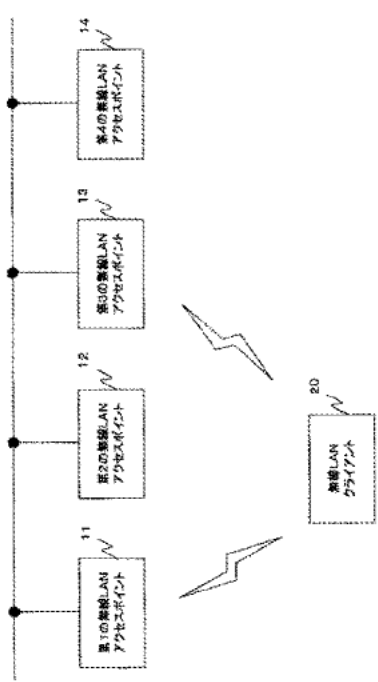
(54) 【発明の名称】 無線LANの高速ローミングシステム、高速ローミング方法及び高速ローミングプログラム

(57) 【要約】

【課題】無線LANネットワークにおいて高速ローミングを実現する。

【解決手段】複数の無線LANアクセスポイントと、当該無線LANアクセスポイントと接続される無線LANクライアントと、を有する無線LANシステムにおいて、前記無線LANアクセスポイントは同一のローカルエリアネットワーク上に存在する無線LANアクセスポイントであって同一のESSIDに設定されている無線LANアクセスポイントの使用チャンネル及びBSSIDを同一ネットワークの無線LANアクセスポイント情報として認識しデータとして保持し、前記無線LANクライアントは前記接続を行う際に前記無線LANアクセスポイント情報を利用する。

【選択図】図1



【特許請求の範囲】**【請求項1】**

複数の無線LANアクセスポイントと、当該無線LANアクセスポイントと接続される無線LANクライアントと、を有する無線LANシステムにおける接続方法において、

前記無線LANアクセスポイントは同一のローカルエリアネットワーク上に存在する無線LANアクセスポイントであって同一のESSIDに設定されている無線LANアクセスポイントの使用チャンネル及びBSSIDを同一ネットワークの無線LANアクセスポイント情報として認識しデータとして保持し、前記無線LANクライアントは前記接続を行う際に前記無線LANアクセスポイント情報を利用することを特徴とする接続方法。

【請求項2】

前記無線LANアクセスポイントは、前記無線LANクライアントと接続された際に、自らが保持している前記無線LANアクセスポイント情報を無線LANクライアントに受け渡すことを特徴とする請求項1に記載の接続方法。

【請求項3】

前記無線LANクライアントは最初に前記無線LANアクセスポイントに接続を試みる際は前記ローカルエリアネットワーク上に存在する前記無線LANアクセスポイントの全チャンネルをスキャンし、当該スキャンをした中から最良の条件の無線LANアクセスポイントを選択し接続し、前記無線LANクライアントが前記無線LANアクセスポイントと圏外から再接続を試みる際、或いは、ローミングする際には、前記受け取った無線LANアクセスポイント情報を基に、最低限のチャンネルをスキャンし、当該スキャンをしたチャンネルの中から最良の無線LANアクセスポイントを選択し接続することを特徴とする請求項1又は2に記載の接続方法。

【請求項4】

前記最低限のチャンネルのスキャンは、前記無線LANクライアントが、前記無線LANアクセスポイント情報の中から自分の接続可能なESSIDを持つ無線LANアクセスポイントのチャンネルを抽出し、当該抽出したチャンネルのみを前記最低限のチャンネルと判断することにより行われることを特徴とする請求項1乃至3の何れか1項に記載の接続方法。

【請求項5】

複数の無線LANアクセスポイントと接続される無線LANクライアントが行う接続方法であって、

前記無線LANアクセスポイントに最初に接続を試みる際は前記ローカルエリアネットワーク上に存在する前記無線LANアクセスポイントの全チャンネルをスキャンし、当該スキャンをした中から最良の条件の無線LANアクセスポイントを選択し接続し、接続された際に前記無線LANアクセスポイントから、当該無線LANアクセスポイントと同一のローカルエリアネットワーク上に存在する無線LANアクセスポイントであって同一のESSIDに設定されている無線LANアクセスポイントの使用チャンネル及びBSSIDを同一ネットワークの無線LANアクセスポイント情報として受け取り、前記無線LANクライアントが前記無線LANアクセスポイントと圏外から再接続を試みる際、或いは、ローミングする際には、前記受け取った無線LANアクセスポイント情報を基に、最低限のチャンネルをスキャンし、その中から最良の無線LANアクセスポイントを選択し接続することを特徴とする接続方法。

【請求項6】

複数の無線LANアクセスポイントと、当該無線LANアクセスポイントと接続される無線LANクライアントと、を有する無線LAN接続システムにおいて、

前記無線LANアクセスポイントは同一のローカルエリアネットワーク上に存在する無線LANアクセスポイントであって同一のESSIDに設定されている無線LANアクセスポイントの使用チャンネル及びBSSIDを同一ネットワークの無線LANアクセスポイント情報として認識しデータとして保持し、前記無線LANクライアントは前記接続を

行う際に前記無線LANアクセスポイント情報を利用することを特徴とする接続システム

【請求項7】

前記無線LANアクセスポイントは、前記無線LANクライアントと接続された際に、自らが保持している前記無線LANアクセスポイント情報を無線LANクライアントに受け渡すことを特徴とする請求項6に記載の接続システム。

【請求項8】

前記無線LANクライアントは最初に前記無線LANアクセスポイントに接続を試みる際は前記ローカルエリアネットワーク上に存在する前記無線LANアクセスポイントの全チャンネルをスキャンし、当該スキャンをした中から最良の条件の無線LANアクセスポイントを選択し接続し、前記無線LANクライアントが前記無線LANアクセスポイントと圏外から再接続を試みる際、或いは、ローミングする際には、前記受け取った無線LANアクセスポイント情報を基に、最低限のチャンネルをスキャンし、当該スキャンをしたチャンネルの中から最良の無線LANアクセスポイントを選択し接続することを特徴とする請求項6又は7に記載の接続システム。

【請求項9】

前記最低限のチャンネルのスキャンは、前記無線LANクライアントが、前記無線LANアクセスポイント情報の中から自分の接続可能なESSIDを持つ無線LANアクセスポイントのチャンネルを抽出し、当該抽出したチャンネルのみを前記最低限のチャンネルと判断することにより行われることを特徴とする請求項6乃至8の何れか1項に記載の接続システム。

【請求項10】

複数の無線LANアクセスポイントと接続される無線LANクライアントにおいて、
前記無線LANアクセスポイントに最初に接続を試みる際は前記ローカルエリアネットワーク上に存在する前記無線LANアクセスポイントの全チャンネルをスキャンし、当該スキャンをした中から最良の条件の無線LANアクセスポイントを選択し接続し、接続された際に前記無線LANアクセスポイントから、当該無線LANアクセスポイントと同一のローカルエリアネットワーク上に存在する無線LANアクセスポイントであって同一のESSIDに設定されている無線LANアクセスポイントの使用チャンネル及びBSSIDを同一ネットワークの無線LANアクセスポイント情報として受け取り、前記無線LANクライアントが前記無線LANアクセスポイントと圏外から再接続を試みる際、或いは、ローミングする際には、前記受け取った無線LANアクセスポイント情報を基に、最低限のチャンネルをスキャンし、その中から最良の無線LANアクセスポイントを選択し接続することを特徴とする無線LANクライアント。

【請求項11】

複数の無線LANアクセスポイントと、当該無線LANアクセスポイントと接続される無線LANクライアントと、を有するシステムに搭載される無線LAN接続プログラムにおいて、

前記無線LANアクセスポイントは同一のローカルエリアネットワーク上に存在する無線LANアクセスポイントであって同一のESSIDに設定されている無線LANアクセスポイントの使用チャンネル及びBSSIDを同一ネットワークの無線LANアクセスポイント情報として認識しデータとして保持し、前記無線LANクライアントは前記接続を行う際に前記無線LANアクセスポイント情報を利用するシステムとしてコンピュータを機能させることを特徴とする接続プログラム。

【請求項12】

前記無線LANアクセスポイントは、前記無線LANクライアントと接続された際に、自らが保持している前記無線LANアクセスポイント情報を無線LANクライアントに受け渡すことを特徴とする請求項11に記載の接続プログラム。

【請求項13】

前記無線LANクライアントは最初に前記無線LANアクセスポイントに接続を試みる

際は前記ローカルエリアネットワーク上に存在する前記無線LANアクセスポイントの全チャンネルをスキャンし、当該スキャンをした中から最良の条件の無線LANアクセスポイントを選択し接続し、前記無線LANクライアントが前記無線LANアクセスポイントと圏外から再接続を試みる際、或いは、ローミングする際には、前記受け取った無線LANアクセスポイント情報を基に、最低限のチャンネルをスキャンし、当該スキャンをしたチャンネルの中から最良の無線LANアクセスポイントを選択し接続することを特徴とする請求項11又は12に記載の接続プログラム。

【請求項14】

前記最低限のチャンネルのスキャンは、前記無線LANクライアントが、前記無線LANアクセスポイント情報の中から自分の接続可能なESSIDを持つ無線LANアクセスポイントのチャンネルを抽出し、当該抽出したチャンネルのみを前記最低限のチャンネルと判断することにより行われることを特徴とする請求項11乃至13の何れか1項に記載の接続プログラム。

【請求項15】

複数の無線LANアクセスポイントと接続される無線LANクライアントに搭載される接続プログラムにおいて、

前記無線LANアクセスポイントに最初に接続を試みる際は前記ローカルエリアネットワーク上に存在する前記無線LANアクセスポイントの全チャンネルをスキャンし、当該スキャンをした中から最良の条件の無線LANアクセスポイントを選択し接続し、接続された際に前記無線LANアクセスポイントから、当該無線LANアクセスポイントと同一のローカルエリアネットワーク上に存在する無線LANアクセスポイントであって同一のESSIDに設定されている無線LANアクセスポイントの使用チャンネル及びBSSIDを同一ネットワークの無線LANアクセスポイント情報として受け取り、前記無線LANクライアントが前記無線LANアクセスポイントと圏外から再接続を試みる際、或いは、ローミングする際には、前記受け取った無線LANアクセスポイント情報を基に、最低限のチャンネルをスキャンし、その中から最良の無線LANアクセスポイントを選択し接続する無線LANクライアントとしてコンピュータを機能させることを特徴とする接続プログラム。

【発明の詳細な説明】

【技術分野】

【0001】

本発明は、無線LANに関し、より詳細には無線LANのローミング又は再接続に関する。

【背景技術】

【0002】

近年無線LANの機器の低価格化等の影響もあり、無線LANが一般に広く普及している。また、無線LANを構築する際に1つのローカルエリアネットワーク内に多数の無線LANアクセスポイントを数設するといったことも一般的に行われている。

【0003】

このような環境下において無線LANクライアントは、1つのローカルエリアネットワーク内の全ての無線LANアクセスポイントを対象として、接続可能な無線LANアクセスポイントを検索する。

【0004】

そして、無線LANクライアントは、検索の結果に基づいて最良と判断した無線LANアクセスポイントに対して接続動作をする（例えば特許文献1参照）。

【図面の簡単な説明】

【0016】

【図1】本発明の実施形態における無線LANネットワークの基本的構成の例を表す図である。

【図2】本発明の実施形態における各機器の設定の例を表す図である。

【図3】本発明の実施形態における無線LANアクセスポイント情報の例を表す図である

。

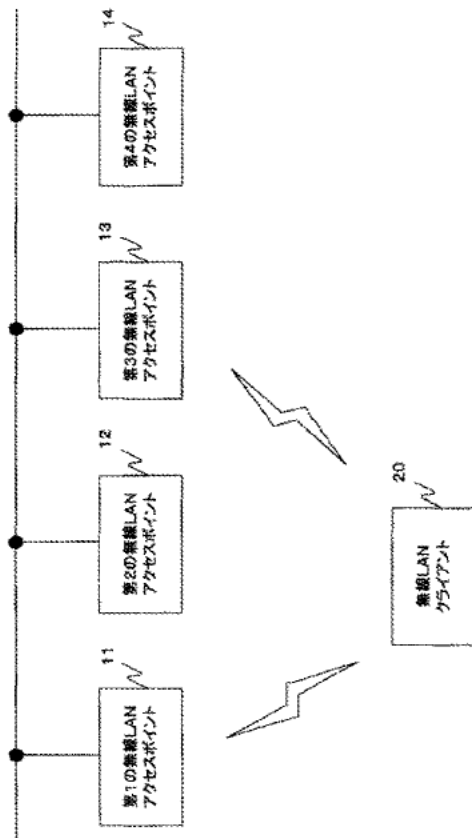
【図4】本発明の実施形態における無線LANクライアントの無線LANアクセスポイント接続動作を表すフローチャートである。

【符号の説明】

【0045】

- 1 1 第1の無線LANアクセスポイント
- 1 2 第2の無線LANアクセスポイント
- 1 3 第3の無線LANアクセスポイント
- 1 4 第4の無線LANアクセスポイント
- 2 0 無線LANクライアント

【図1】



各機器の設定			
SYSTEM NAME	BSSID	ESSID	CH
AP11	11111-11111	Office1	1
AP12	22222-22222	Office1	4
AP13	33333-33333	Office1	7
AP14	44444-44444	Office2	14
CL20	—	Office1	—

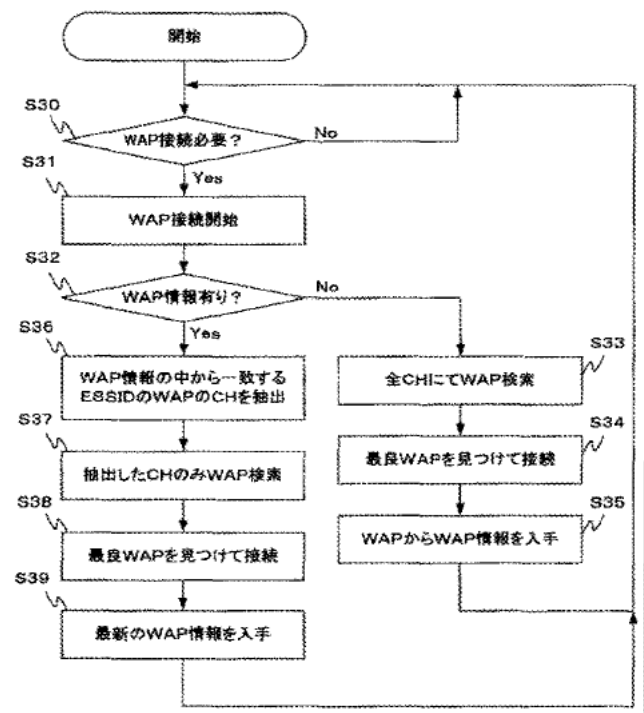
無線LANアクセスポイント情報			
SYSTEM NAME	BSSID	ESSID	CH
AP11	11111-11111	Office1	1
AP12	22222-22222	Office1	4
AP13	33333-33333	Office1	7
AP14	44444-44444	Office2	14

【※3】

(8)

特開2011-4225 (P2011-4225A)

【図4】



(19) World Intellectual Property Organization
International Bureau(43) International Publication Date
9 January 2003 (09.01.2003)

PCT

(10) International Publication Number
WO 03/003760 A2(51) International Patent Classification⁷: **H04Q 3/00**(74) Agents: **SWISS, Gerald, F. et al.**; BURNS, DOANE, SWECKER & MATHIS, L.L.P., P.O. BOX 1404, Alexandria, VA 22313-1404 (US).

(21) International Application Number: PCT/US02/16590

(22) International Filing Date: 25 June 2002 (25.06.2002)

(84) Designated States (regional): European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR).

(25) Filing Language: English

(26) Publication Language: English

Published:

— without international search report and to be republished upon receipt of that report

(30) Priority Data:

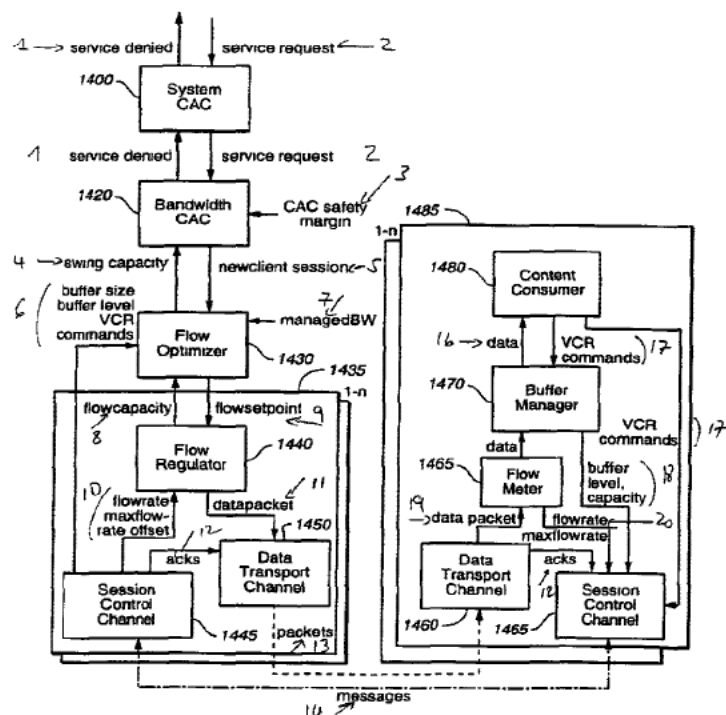
09/893,364

26 June 2001 (26.06.2001) US

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

(71) Applicant: **BURST.COM** [US/US]; 613 Fourth Street, Suite 201, Santa Rosa, CA 95404 (US).(72) Inventor: **ALLEN, Arthur, Douglas**; 1322 Isabelle Avenue, Mountain View, CA 94040 (US).

(54) Title: METHOD FOR CONNECTION ACCEPTANCE CONTROL AND RAPID DETERMINATION OF OPTIMAL MULTI-MEDIA CONTENT DELIVERY OVER NETWORKS



(57) Abstract: A method of bandwidth allocation for delivery of stored digital content from at least one server device to at least one client device by way of a network is disclosed. The method begins by prescribing a control variable which represents a target flow rate from the server device to each client device. Next, time-varying constraints on the flow rate of the content are determined. A cost function of the control variable for each client is determined. The cost function corresponds to a maximized value of the control variable. Finally, bandwidth is prescribed to each client based upon the value of the control variable maximized by the cost function. In this respect, the method achieves optimal allocation of bandwidth between the server and the respective clients.

WO 03/003760 A2

-1-

**METHOD FOR CONNECTION ACCEPTANCE CONTROL AND RAPID
DETERMINATION OF OPTIMAL MULTI-MEDIA CONTENT
DELIVERY OVER NETWORKS**

BACKGROUND OF THE INVENTION

Field of the Invention

[0001] This invention relates to the field of delivery of multimedia content over a variety of networks. More specifically, it pertains to multimedia servers which service many clients simultaneously for the delivery of multimedia content which is used and played back at each client. It addresses methods for determining optimal delivery rates to each client and methods for determining whether new clients may be accepted without diminishing the quality of service to existing clients.

Status of the Prior Art

[0002] In the history of multimedia program delivery, some in the industry have long advocated the use of large client-side buffers and faster-than-real-time content delivery over a network as offering the best of all worlds: a jitter-free viewing experience and a cost-effective utilization of the network resources at hand. Few systems, however, go very far in addressing how to schedule clients or a method for accepting new clients. Real-time systems, often known as streaming systems, can schedule new clients in a very simple manner -- if sufficient bandwidth remains for the added real-time stream, then the client may be accepted. However, such systems do not maximize the number of simultaneous clients. On the other hand, faster than real-time delivery, sometimes known as store-and-forward systems, opens up the possibility for more flexible scheduling procedures to control and optimize the number of simultaneous clients while ensuring a high level of quality of service.

-2-

[0003] The methods for such call acceptance and flow modulation that have been proposed in the prior art have been largely ad-hoc and also incomplete. These have been ad-hoc in the sense that there has been no guiding rationale for their selection from among many possible and potentially superior alternatives. The methods have also been incomplete insofar as they did not address the question of whether any given incoming request for service should be accepted or denied. Video-on-demand systems, or more generally, any system in which a multimedia server is designed to serve multiple clients over a network to deliver bounded content, can benefit from the use of such flow modulation techniques and call acceptance procedures.

Optimal Content Flow Modulation

[0004] One time-honored way of designing methods of the class required here is to re-cast the problem to be solved as an optimization problem, in which one seeks to maximize a designated value function moment-by-moment, subject to a set of real-world operational constraints which will typically vary over time. Accordingly, given a set of clients and associated sessions, an optimal delivery procedure continuously establishes content flow rates from the content server to each of its clients so as to maximize aggregate value according to the governing value function.

[0005] This approach holds several advantages: 1) optimization problems are well understood, and are tractable by a large and diverse collection of computational methods; 2) if it exists, a global solution that is obtained is arguably optimal by construction, and thus superior or equal to all other.

-3-

[0006] The present invention teaches the method of optimizing two particular value functions:

- 1) total data delivered (maximize throughput).
- 2) total delivery charges (maximize charges).

5 [0007] The first value function does not distinguish one customer from another and will deliver as much data as possible from server to clients irrespective of the characteristics of the latter. The second value function favors the service of high paying customers. It can easily be seen that the first function is a special case of the second one whereby all clients are charged equally.

10 [0008] As will be seen in this disclosure, optimizing for these functions and identifying the necessary constraints requires a new and unique perspective that is specifically designed for the multimedia environment. Moreover, the disclosed methods are specifically designed to account for and accommodate real-world scenarios of today's networks. Consequently many variations of the method are
15 presented to accommodate various scenarios.

SUMMARY OF THE INVENTION

Call/Connection Acceptance Control (CAC)

[0009] A CAC procedure is responsible for deciding whether a candidate for service can be accommodated without jeopardizing sessions already in progress at
20 the present time or at some time in the future; failing that it must decide whether a service request should be queued for a time or rejected.

Flow Modulation

[0010] Flow modulation methods are those portions of the system which manage the communication and data flow between the server and the clients. Collectively, these methods provide the multimedia data to the client and provide the server with the information about the state of the transmission, playback, user status and network status. These parameters are further used by the present invention in the CAC procedures. In fact, as will be shown, the proposed CAC procedures are tightly integrated with the flow modulation methods.

Adaptation to Variations in Network Capacity

[0011] Operational constraints may change over time. For instance, one might elect to vary the total bandwidth available for multimedia content delivery according to the time of day. Alternatively, exogenous data flows on the network may cause unexpected disturbances by usurping available bandwidth and impeding the delivery of data along established session channels. The content delivery strategy of the present invention includes the ability to adapt to scheduled as well as unexpected disturbances so as to minimize unwanted disruptions of services.

Burst Transmissions Provide the Opportunity to Adapt

[0012] The present invention, due to its faster-than-realtime transmissions (also known as burst transmissions), which are realized by use of high-bandwidth networks and large client cache or intermediate storage, provides an opportunity to adapt to changing network conditions. In contrast real-time (streaming) systems are essentially designed for worst-case scenarios: each client must be assumed to constantly use the complete real-time playback bandwidth. Such a system is unable to adapt to any derivation from this scenario. For example, take the simple case where the total server bandwidth is 100% utilized by all clients playing back the streaming video. Should any network condition change, such as a temporary

-5-

decrease in available bandwidth over the network, then one or more clients' playback is interrupted, and the system can not recover from such a condition until the bandwidth is regained. Even worse if a single client presses pause either that unused bandwidth must remain reserved and no more clients can be accepted, or
5 that paused client is pushed out in order to service the new client. In essence little or no CAC procedure may be implemented.

[0013] In contrast the present invention burst transmits portions of a program and immediately 'gets ahead of itself', thus allowing headroom for a myriad of methods to intelligently handle new clients, client interactivity and possible
10 network fluctuations.

[0014] Methods are taught for optimally determining the flow rate to each client. Methods are also taught for accepting or rejecting new clients; these call-acceptance methods are tightly coupled with said flow rate modulation methods. A series of constraint expressions are presented which govern the methods for
15 determining the flow rates and acceptance of new clients. Linear programming techniques are used to optimally solve these expressions. Various embodiments are presented including scenarios for multiple-rate tariffs, and time-of-day bandwidth variations.

BRIEF DESCRIPTION OF THE DRAWINGS

[0015] These as well as other features of the present invention will become more apparent upon reference to the drawings wherein:

5 Figure 1 depicts the flow of control and/or data between the different stations of a content delivery session;

 Figure 2 illustrates the Entity Data Model;

 Figure 3 geometrically illustrates the linear programming problem statement;

10 Figure 4a geometrically illustrates an expansion of the flow optimization problem statement;

 Figure 4b geometrically illustrates the method for rapid determination of flow rates to maximize flow;

 Figure 5 illustrates a method for implementing flow modulation to maximize flow;

15 Figure 6 illustrates a method for implementing flow modulation for maximized charges;

 Figure 7 illustrates typical content flow;

 Figure 8 illustrates typical server swing capacity;

 Figure 9 illustrates a method for call-acceptance and control (CAC);

20 Figure 10 illustrates planned constraints on maximum flow;

 Figure 11 illustrates a method for call-acceptance and control (CAC) with scheduled flow changes;

 Figure 12 illustrates stratification of services;

25 Figure 13 illustrates a method for call-acceptance and control (CAC) for maximal charge;

 Figure 14 is a system block diagram for a system implementing the present optimization of client bandwidth;

-7-

Figure 15 is a flowchart showing an algebraic method for calculating the optimization of client bandwidth;

Figure 16 is a flowchart showing an algebraic method for calculating the optimization of client costs; and

5 Figure 17 illustrates optimized viewing of a live event.

DETAILED DESCRIPTION OF THE INVENTION

Data & Control Flows (Figure 1)

[0016] Figure 1 depicts the flow of control and/or data between the different stations of a content delivery session. As shown a client attempts a connection 100 and manifests itself to the Content Selection subsystem by means of a low bandwidth control channel (not shown). Next the client is authenticated and a selection is made 110, typically with the aid of browser software. If the client is not authenticated, it is dismissed from the system 120. If the client has been authenticated and a program selected for viewing then the rate of service is set at this point 130, perhaps according to the selection that was made, or some contractual stipulation. The client is now placed on the service queue of the CAC subsystem 140. A client that is made to wait too long will eventually balk 150. Assuming this does not occur, the CAC subsystem 140 will eventually allocate a channel to the client and open a session 160. Control now devolves upon the Content Flow Modulator (not shown) which starts the flow of content from server to client 170. Subsequent capacity changes, whether predictable or not, may force an abrupt termination of a session in progress 180. Otherwise the session runs to completion 190.

-8-

Entity Data Model (Figure 2, listing, table)

[0017] The entities entering into our discussion are depicted in Figure 2. Client 200 maintains certain data associated with this entity; as shown but not labeled, which includes without limitation, status, id and costOfService. The other entities also each include unlabeled but depicted data. The diagram further depicts the relationship between each entity. As shown, client 200 is assigned a session 240. Client 200 employs a channel 210. Client 200 selects contentSelection 230. Session 240 delivers content through channel 210. Server 220 modulates channel 210. Server 200 contains contentSelection 210. Server 220 accepts, defers or denies client 200 and contentSelection 230 is associated with session 240.

[0018] Furthermore Figure 2 depicts the various one-to-many relationships. Each client 200 employs one channel 210. Client 200 may or may not receive one of channel 210, as depicted by the 0/1 notation. Similarly, client 200 may or may not receive a session 240. However, whenever client 200 does receive a session 240, it will always receive a channel 210 since channel 210 and session 240 are allocated as a pair. One or more (N) of client 200 may select one of contentSelection 230. And server 220 contains one or more (N) of contentSelection 230. Each one of contentSelection 230 is associated with 0-N of session 240. Each session 240 delivers content through one of channel 210. And server 220 modulates one or more (N) of channel 210.

[0019] A more detailed list of each entity of Figure 2, and each one's associated description, data elements and function calls is listed below. This listing closely resembles that of object-oriented programming. As such, 'methods' represent the ability to obtain or modify data, while 'attributes' represent data which is directly associated with that particular entity. The listing also includes information relating to one embodiment wherein software programming specifics are disclosed, such as

-9-

a variable type (double, int and so forth) and more. The present invention is not limited to such an embodiment and other implementations are possible without deviating from the scope and intent of the present invention. The listing, however detailed, is merely illustrative of the data and functions which are used in the
5 equations and methods described herein.

[0020] Consequently, data and functions from this listing, associated with the various entities, will be used in forthcoming equations, flowcharts and methods. The reader is directed to this listing as reference when reading such equations and examining such drawings.

10 --- start of entity data model detailed listing ---

Model: Untitled 1 (public)

Contains:

client, session, channel, server, contentSelection.

Component: client (public Class/Interface)

15 *Comment:*

A client entity stands for a client presently requesting or receiving service.

Methods:

public static lookup (id: in int) : client

public GetId () : const int&

20 public SetId (val : in int&)

public GetCostOfService () : const double&

public SetCostOfService (val : in double&)

-10-

*Attributes:***private status: client < int >**

Specifies whether or not a client has been allocated a channel and session.

private id: int

- 5 Integer-valued identifier that is unique to the client (primary key). Can be obtained from a monotonically increasing counter.

private costOfService: double

- 10 Dollar charge per Mbyte. This value is the same for all customers under flow optimization. Under cost/charge optimization may be an integer value reflective of the rank; the higher the rank the higher the charge.

*Has:***public selected: contentSelection****public assigned a: session****public employs: channel**

- 15 **Component: session (public Class/Interface)**

Comment:

A session entity holds various state information about the service being received by an associated customer.

- 20 **public GetCurrentPosition () : const double&**
public SetCurrentPosition (val : in double&)
public GetPayloadToGo () : const double&
public SetPayloadToGo (val : in double&)
public GetStatus () : const int&

-11-

```

public SetStatus (val : in int&)
public GetMinimumFlowRate () : const double&
public SetMinimumFlowRate (val : in double&)
public GetFlowRateRange () : const double&
5 public SetFlowRateRange (val : in double&)
public GetMaxFlowRate () : const double&
public SetMaxFlowRate (val : in double&)

```

Attributes:

```

private playTimeToGo: double
10 Indicates the minutes remaining in the viewing experience. Initialized to
contentSelectiont.playTime (see below).

private currentPosition: double
Pointer into media content from which content is being delivered.

private payloadToGo: double
15 The amount of media content (in Mbytes) as yet undelivered by the server. Does
not include any content presently stored in the client-side buffer.

private status: int
Indicates whether session is active or paused.

private minimumFlowRate: double
20 This is the minimum flow from server to client required to ensure uninterrupted
service over the remaining playTime. Has a value of zero if payloadToGo is
zero. Given by (payloadToGo * 8)/(playTimeToGo * 60)

```

-12-

private flowRateRange: double

Specifies the effective range over which the channel content flow serving a session is constrained without consideration for interactions with other flows. Equals $\text{maxFlowRate} - \text{minimumFlowRate}$

5 **private maxFlowRate: double**

Effective maximum bound on flow as expressed in formula (8) which must be re-evaluated periodically.

Has:

public delivers content through: channel

10 **Component: channel (public Class/Interface)**

Comment:

A channel represents the network resources from server to client associated with an ongoing session, encompassing the client-side buffer if any, and its level.

15 **public GetBufferLevel () : const double&**

public SetBufferLevel (val : in double&)

public GetFlowRate () : const double&

public SetFlowRate (val : in double&)

public GetMaxFlowRate () : const double&

20 **public SetMaxFlowRate (val : in double&)**

Attributes:

private bufferSize: double

Capacity of the client-side buffer (or equivalent)

-13-

private **bufferLevel: double**

Current buffer level in MBytes of stored content.

private **flowRate: double**

Flow rate through channel specified by the relevant optimizing flow
modulator.

private **maxFlowRate: double**

This value represents the maximum possible flow rate from the server to an individual client over its "channel". This value reflects restrictions on flow that pertain to an individual client. It may be determined by factors such as the bandwidth of client's link to the network, or a limit imposed administratively to ensure balanced network utilization.

Component: server (public Class/Interface)

Comment:

Entity representing the media server and its CAC and flow modulation activities.

public **GetFlowRate () : const double&**

public **SetFlowRate (val : in double&)**

public **GetMaxMinFlowRate[] () : const double&**

public **SetMaxMinFlowRate[] (val : in double&)**

Attributes:

private **maxFlowRate: double**

Maximum possible content flow that is allocated to the server by the network.

private **flowRate: double**

Aggregate content flow rate, summed over all sessions and their associated channels.

-14-

private cac_flowSafetyMargin: double

Tunable safety margin used by the CAC algorithm to protect sessions-in-progress from being affected by changes in available network bandwidth.

private maxMinFlowRate[]: double

- 5 Applies when N rate tariffs exist. This array holds the maximum floor level for each category of service. The value for the costliest category N is stored in maxMinFlowRate[N-1], and for the least costliest in maxMinFlowRate[0]. It is the relative magnitude of these ascending values that matters, not their absolute value. Thus the actual maximum floor flow rate for category k is
- 10 given by $\text{server.maxFlowRate} * (\text{server.maxMinFlowRate}[k-1] / \text{server.maxMinFlowRate}[N-1])$. Similarly, the maximum floor flow rate for category N is server.maxFlowRate.

Has:

public contains: contentSelection

- 15 **public modulates: channel**

Component: contentSelection (public Class/Interface)

Comment:

Entity represents a video/sound clip or other bounded unit of content. A continuous data feed does not qualify.

- 20 *Attributes:*

private averagePlayRate: double

The average rate at which media content is consumed by the client, as computed by dividing the (payload * 8) by the (playTime * 60)

-15-

private playTime: double

Duration of play of the media content in minutes.

private **payLoad:** double

total size of the content in Mbytes.

5 *Has:*

public is associated with: session

--- end of entity data model detailed listing ---

The following table summarizes the highlights of the previous detailed description of each entity in Figure 2.

10

TABLE 1

Entity	Description
client 200	Each client is denoted by an associated unique integer index_{Id}. The set of active clients is denoted by $S_{ActiveClients}$. The set of deferred clients is denoted by $S_{QdClients}$. Incoming clients are expected to select the content they wish to view prior to being queued for dispatch by the CAC sub-system, which requires knowledge of the client's bandwidth requirements, duration of play, and cost of service, all of which may vary according to the selection.

-16-

- server 220 Servers sit astride a network and can deliver media content through the network to their clients up to a designated maximum flow rate. The server is responsible for accepting or rejecting clients, launching sessions and associated channels for the former, and modulating content flows over all channels in an optimal manner.
- channel 210 A channel represents the data path between the server and the client. The channel buffer is typically located near or within the clients viewing station. The flow of content through the channel is set by the flow modulator sub-system.
- contentSelection 230 A server will typically act as a repository for media content, which it can deliver to clients upon demand. For our purposes, media content is characterized by its payload and the play duration, which together imply the *averagePlayRate* = (payload*8)/(playTime *60). The *averagePlayRate* is none other than the streaming rate imposed by real-time just-in-time streaming algorithms.
- session 240 Every session represents an instance of media content delivery to an associated client over a designated channel. The *playTimeToGo* indicates the time remaining before the content is fully played out to the client. The *payloadToGo* is the amount of content data as yet undelivered to the channel. A session terminates when this value reaches zero, at which time *playTimeToGo* may still be large, according to the capacity, the level of the channel buffer, and the media play rate.

-17-

Constraints On Content Flow

[0021] Before referring to more Figures, it is imperative to establish some formulas and problem statements which are used in the methods which follow.

5 [0022] The flow of content between entities is subject to the following constraints at all times. Buffer levels are always expressed in Mbytes and data rates in Mbits/sec.

(1) $\sum_{i \in \text{SactiveClients}} (\text{client.lookup}(i).\text{channel.flowRate}) \leq \text{server.maxFlowRate}$

The sum of all channel flows cannot exceed the imposed maximum throughput capacity of the server.

10 (2) $\text{client.lookup}(i).\text{channel.flowRate} \leq \text{client.lookup}(i).\text{channel.maxFlowRate}$
for all $i \in \Sigma_{\text{activeClients}}$

The data path from server to client is subject to its own constriction.

(3) $\text{client.lookup}(i).\text{channel.flowRate} \leq (\text{client.lookup}(i).\text{channel.bufferSize} - \text{client.lookup}(i).\text{channel.bufferLevel}) * 8/60 +$
15 $\text{client.lookup}(i).\text{session.mediaContent.averagePlayRate}$, for all $i \in \Sigma_{\text{activeClients}}$,

The channel buffer is never allowed to overflow.

(4) $\text{client.lookup}(i).\text{channel.flowRate} \leq \text{client.lookup}(i).\text{session.payloadToGo} * 8/60$

-18-

for all $i \in \Sigma_{\text{activeClients}}$,

Content that does not exist cannot be delivered. (Constraint 1 will ordinarily prevail except at the very end of a session.)

[0023] The constraints listed above are straightforward applications of common sense in relation to the flow of data through constricted channels, out of finite data sources, and into and out of bounded buffers. By contrast, the following constraint, which imposes a minimum channel flow rate instead of a maximum, is less obvious. The minimum value, termed the *minFlowRate* is set to the flow rate which, if sustained over the balance of the play time to go (*playTimeToGo*), ensures that all required content will be available when needed -- and no sooner -- until all content is played out. This floor value can be calculated for $i \in \Sigma_{\text{activeClients}}$ by the formula:

$$(5) \text{client.lookup}(i).\text{session.minFlowRate} = (\text{client.lookup}(i).\text{session.payloadToGo} * 8) / (\text{client.lookup}(i).\text{session.playTimeToGo} * 60)$$

Accordingly, the flow rate on the right-hand-side is termed the just-in-time (JIT) flow rate (f^{JIT}).

Thus:

$$(6) \text{client.lookup}(i).\text{channel.flowRate} \geq \text{client.lookup}(i).\text{session.minFlowRate} \\ \text{for all } i \in \Sigma_{\text{activeClients}}$$

[0024] The variable constraint bounds (i.e. the values to the right of the inequality symbol) of equations 1 – 4 and 6 are re-evaluated on a periodic basis (e.g. once per second) prior to the execution of the CAC procedure and optimizer. In

-19-

particular, the *minFlowRate* value starts out at the beginning of a session equal to the streaming rate. By construction the *minFlowRate* rate never exceeds this initial value so long as constraint 6 is honored. In fact, constraint 5 implies that the *minflowRate* value must be a diminishing function of time that may hold its value for a time but never rises. As seen from equation 6, the actual data rate of the channel, *flowRate*, is always greater than or equal to the *minFlowRate*. By design, and virtue of the fact the present invention uses faster-than-realtime transmissions, the system quickly gets ahead of itself and ensures that after initial conditions, the *minFlowRate* is always equal to or less than the real-time rate and that it continues to decrease. As we shall see, the CAC procedure exploits this monotonic characteristic of the minimum flow rate over time.

[0025] Constraints 2, 3 and 4 are of like kind, each specifying an upper bound on individual channel flows. Whereas the bound for constraint 2 is typically a constant, the bounds on 3 and 4 will vary over time. Nevertheless, only one of the three bounds is effective at any given time, namely the one with the smallest bound value, given by:

(7) $\text{client.lookup}(i).\text{session.maxFlowRate} = \text{minimum of}$

1) $\text{client.lookup}(i).\text{channel.maxFlowRate},$

2) $(\text{client.lookup}(i).\text{channel.bufferSize} -$
 20 $\text{client.lookup}(i).\text{channel.bufferLevel}) *$

$8/60 + \text{client.lookup}(i).\text{session.mediaContent.averagePlayRate},$

-20-

$$3) \text{ client.lookup(i).session.payloadToGo} * 8/60$$

Consequently, formulas 2, 3, and 4 can be consolidated into a single constraint, the bound for which is computed at every scan to be the smallest bound of associated constraints 2, 3 and 4.

$$5 \quad (8) \text{ client.lookup(i).channel.flowRate} \leq \text{client.lookup(i).session.maxFlowRate},$$

whereby for all $i \in \Sigma_{\text{activeClients}}$, maxflowRate is given by equation (7).

At any one time, individual channel flows are constrained over a range, as follows:

$$10 \quad (9) \text{ client.lookup(i).session.flowRateRange} = \text{client.lookup(i).session.maxFlowRate} - \text{client.lookup(i).session.minimumFlowRate}$$

Value Functions

[0026] The value functions introduced previously can be expressed mathematically as linear functions of channel flows, as follows:

Optimizing Throughput (Maximal Flow)

$$15 \quad (10) \text{ value} = \sum_{i \in \text{SactiveClients}} \text{client.lookup(i).channel.flowRate}$$

Optimizing Charges (Maximal Charges)

$$(11) \text{ value} = \sum_{i \in \text{SactiveClients}} (\text{client.lookup(i).channel.flowRate} * \text{client.lookup(i).costOfService})$$

-21-

Optimization Problem Statement (Figure 3)

[0027] The optimization problem, which in one embodiment is strictly throughput and in another case is charge, can be stated simply as follows:

Find values for

5 client.lookup(i).channel.flowRate for all $i \in \Sigma_{\text{activeClients}}$

constrained by inequalities 1 through 5, such that the value obtained by evaluating expression 10 or 11 assumes a maximum.

[0028] Both of these problem formulations are examples of Linear Programming for which a number of well-known and generally effective computational solutions exist. In linear programming one seeks to optimize a linear cost function of variable x

10

$$(12) \quad c * x = c_1 * x_1 + \dots + c_n * x_n$$

subject to a set of linear inequality constraints

$$(13) \quad A * x \leq b$$

15 where $x^T = (x_1, \dots, x_n)$, $c = (c_1, \dots, c_n)$ are the state variable & cost vectors, A is an n -by- m matrix, $b^T = (b_1, \dots, b_m)$ is the constraint vector, and the operator '*' stands for matrix or scalar multiplication.

[0029] Figure 3 is introduced as illustrative of the problem statement and the general methods of the prior art, and is not incorporated as an element of the invention.

20

-22-

- [0030] The linear programming problem as well as its solution can best be understood with the aid of geometry. Figure 3, depicting a 2-dimensional Cartesian problem space, inequality constraints (13) define a convex hull H 310 over which a search for an optimum value of $x = (x_1, x_2)$ is permitted to range.
- 5 The cost vector c 350 defines an infinite family of equal cost lines (hyper-planes) which lie orthogonal to c . Three examples of such lines are shown in L_1 360, L_2 365, and L_3 370., each of progressively higher value. The supreme value of the cost function is obtained by sliding along c 350 till one can go no further, in this instance toward vertex V_4 340 of hull H 310. Many well-known methods (e.g. the
- 10 Simplex Method) work roughly in this fashion, exploiting the fact that at least one optimum point must be at a vertex. In particular, the Simplex method algorithm begins by finding a vertex (e.g. V_2 320), and then moves along a sequence of vertices (e.g. V_3 330, V_4 340) improving itself each time until no further improvement is possible & the summit is reached.
- 15 [0031] Let us suppose instead that V_3 330 were placed along L_3 370 along with V_4 340. According to prior art methods, V_3 330 and V_4 340 are the two possible solutions, but the equally valuable points in between them are not. As we shall soon see, the problem of throughput optimization (6) falls in this category.

- [0032] While vertex V_1 300 does not factor into this description, it is depicted in
- 20 Figure 3 for completeness.

Flow Modulation

Methods for Maximal Flow

The following sections detail two embodiments to optimize total data flow.

Overview (Figure 4-a)

[0033] Figure 4-a depicts a scenario involving two flows. The convex hull is in this instance bounded by line segments L1, L2, L3, L4 and L5. L6 is a boundary used in a different embodiment, however the present embodiment uses L5 and not L6. Flow f_2 can range over the interval separating line segments L1 from L3, namely f_2^{MIN} and f_2^{MAX} ; the range is depicted as f_2^{RANGE} . Flow f_1 can range over the interval between lines L2 and L4, namely f_1^{MIN} and f_1^{MAX} , and depicted as f_1^{RANGE} . The sum of flows f_1 and f_2 is constrained to lie inside of line segment L5 which, by construction, is always orthogonal to the cost vector C_r . Cost vector C_c is also illustrated but used in a distinct embodiment. In the present embodiment only C_r is used. In the illustrated example of the present embodiment, the constraint on total flow is set to 5, and is therefore low enough to cause L5 to intersect L3 and L4. This would not have been true had the value chosen had been 10 instead of 5. With L5 safely out of contention, the convex hull would instead be a simple rectangle bounded by L1 through L4, thereby permitting both flows to assume their respective maxima without interference. In practice operational constraints exist intrinsically or are imposed from the outside so as to ensure cost effective sharing of potentially costly network resources.

[0034] Supposing Figure 4 to be correct, the well-known methods would select vertex V_{3-5} , which lies at the intersection of L3 and L5, or V_{4-5} , which lies at the intersection of L4 and L5. These solutions, though optimal, are undesirable for the present invention as they fail to spread available bandwidth over all channels as fairly as would a centrally located interior point of L5. For this reason, two alternative optimization methods are taught, which are adapted to the particular needs of this problem and ensure a fairer allocation of constrained bandwidth among all channels.

Iterative Procedure (Figure 5)

[0035] In order to optimize use of all available bandwidth, the following general method is used, with the details illustrated in Figure 5. This method is a solution for the problem illustrated in Figure 4-a, which geometrically illustrates the optimization problem in the limited case of two flows, f_1 and f_2 . The following description expands the problem to an arbitrary number of clients (and therefore flows) and presents a method for solving this optimization problem.

[0036] Referring to Figure 5, in step 500, values are calculated for the *session maxFlowRate* and *session.minFlowRate* for each client as per the minimum and maximum constraint bound expressions in equations 6 and 8, respectively. This step correlates to the determination of f_{1min} , f_{1max} , f_{2min} and f_{2max} from Figure 4-a.

[0037] The difference between these two yields the *session.flowRateRange* of each client. Thus:

15 *session.flowRateRange* = *session maxFlowRate* - *session.minimumFlowRate*

[0038] In step 505, the active clients are sorted in an ascending fashion based upon their *session.flowRateRange*. As will be shown this critical step facilitates allocation of the remaining server bandwidth as evenly as possible among all active channels, thus maximizing the number of channels that benefit by use of the total server bandwidth. An arbitrary assignment of remaining bandwidth is likely to saturate the server before all channels have been assigned extra bandwidth, thereby favoring certain channels on an ad-hoc basis. This step correlates to keeping the solution in bounds of the space delineated by line segments L1-L5 of Figure 4-a.

-25-

[0039] In step 510, each client's channel flow rate is set to the *session minimumFlowRate*.

[0040] By doing so, it is ensured that the minimum flow constraint is met for each session and that the minimum flow rate is a non-increasing function of time, which
 5 is critical to the proper functioning of the CAC procedure. This portion of the process also ensures that the solution, starting from the vertex in Figure 4-a as defined by the intersection of L1 and L2, moves generally in the direction of vector c_f . All clients are also marked as unprocessed.

[0041] In the next step, 520, *server.flowRate* is set to the sum of each active
 10 client's *session.flowRate*.

[0042] Next, the following is iterated over all clients in sorted sequence (during any given iteration the selected client is given by its *id*) by performing steps 530 through 570. In step 530 evaluating the following expressions test for possible server saturation:

15 $\text{delta} = (\text{server.maxFlowRate} - \text{server.flowRate}) / (\text{qty of un-processed clients})$
 $\text{range} = \text{client.lookup(id).session.maxFlowRate} -$
 $\text{client.lookup(id).session.flowRate}$

[0043] If *range* is greater than *delta*, this implies that the server can be saturated in this iteration by allocating *delta* to all unprocessed clients (step 540).

20 [0044] On the other hand, the 'no' path for step 530 implies that the server is not saturated and that the present client (given by *id*) will saturate first. Accordingly, in 550 the *delta* variable is set as follows:

$\text{delta} = \text{range}$

-26-

[0045] To again correlate this process back to the geometry of Figure 4-a, server saturation is indicated when the solution which is being sought in the direction of vector c_f goes beyond line segment L5.

5 [0046] Next, the flow rate is incremented for all unprocessed clients by *delta*, causing client *id* to saturate.

[0047] In step 560 the server flow rate is adjusted accordingly:

server.flowRate = server.flowRate + delta * (qty of unprocessed clients)

In step 570 the client given by *id*, now saturated, is marked as processed.

Algebraic Procedure (Figures 4-b and 15) By Interpolation

10 [0048] Referring to Figure 4-b, the iterative method described in the previous section begins its search for a maximum from vertex $V_{\min}$ along the direction shown by line C_1 . In contrast, the present method follows the diagonal line segment L_{diag} of the rectangle bounded by lines L1 through L4, starting at vertex $V_{\min}$ and ending at point V_{3-4} .

15 [0049] Provided it exists, the intersection between L_{diag} and L5, indicated by V_f , is optimal on the same basis as any other point lying along the intersection of L5 with the rectangle bounded by L1 through L4, as previously discussed.

[0050] Whenever L5 does not intersect this rectangle, the optimal solution is given by vertex V3-4 at which f_1 and f_2 both assume their respective maxima.

-27-

[0051] In the first instance, the coordinates for point V_f can be obtained by elementary vector geometry and algebraic manipulation, as follows:

First, we define a number of abbreviations,

- 5 $f_k^{\min}$ represents client.lookup(k).session.minimum flow rate
 f_k^{range} represents client.lookup(k).session.flow rate range
 $f_{\text{svf}}^{\max}$ represents server.maxflowrate

We seek scale factor x such that the vector sum of

$(f_1^{\min} \dots f_n^{\min}) + x (f_1^{\text{range}} \dots f_n^{\text{range}})$ intersects an equi-cost hyperplane for capacity c .

The point of intersection is obtained by solving equation:

10
$$(f_1^{\min} + \dots + f_n^{\min}) + x(f_1^{\text{range}} + \dots + f_n^{\text{range}}) = c$$

Solving for x , we obtain:

$$x = [f_{\text{svr}}^{\max} - (f_1^{\min} + \dots + f_n^{\min})] / (f_1^{\text{range}} + \dots + f_n^{\text{range}})$$

where $f_{\text{svr}}^{\max} - (f_1^{\min} + \dots + f_n^{\min})$ represents the unused bandwidth beyond the minimum allocation for each session. For any given session, the optimal flow rate is given

15 by:

$$f_i = f_i^{\min} + x f_i^{\text{range}}, \text{ if } x < 1,$$

or

$$f_i = f_i^{\min} + f_i^{\text{range}}, \text{ if } x > 1$$

[0052] This solution by interpolation is interesting by virtue of the great efficiency of the calculation, and the fact that no session saturates ($x > 1$) unless all sessions saturate. In contrast, the previous method entails considerably more computational effort, and tends to saturate sessions with the lowest reserve capacity.

20

-28-

[0053] A block diagram for this algorithm is depicted in Figure 15. Block 1500 establishes the maxim and minima for each flow. For each flow the minimum is subtracted from the maximum to obtain the range. Available Bandwidth is calculated in block 1520, as the difference between the aggregate flow capacity and the sum of flow minima. Factor x is calculated next in block 1540, and tested against 1 in block 1650. Flow rates for each session are computed in block 1560 if x is less than 1, or 1570 in the contrary case.

A Method for Maximal Charge

[0054] The following sections detail one embodiment to optimize the total monetary charges within the system. The second method is algebraic in nature, and thus very efficient.

Overview (Figure 4-a)

[0055] Referring back to Figure 4-a, cost vector C_c lies orthogonal to line L6, which intersects the convex hull at the vertex formed by the intersection of lines L4 and L5, namely $V_{4,5}$. This cost vector, and the optimal point that it implies, favors flow f1 over flow f2. In this example, as the cost of service for f1 equals 2, thus exceeding the cost of service of 1 set for f2. As the number of flows grow to exceed the number of distinct categories of service (and associated costs of service) the unique optimal solution, depicted in Figure 4 for the case where every flow has a distinct cost of service, no longer applies. Once again a plurality of flows within a service category vie for bandwidth which a method should endeavor to distribute evenly. This method is derived from the previous one, and optimizes one cost category after another, starting with the most costly and ending with the least costly, or when all available capacity is allocated.

An Iterative Search Procedure (Figure 6)

[0056] Let the service categories be denoted by $k = 1 \dots N$, where k also denotes the cost of service.

5 [0057] Let $C_1 \dots C_N$ be the partition of $S_{\text{activeClients}}$ that places all clients with cost of service k in set C_k . Partition sets C_k can be ordered to form sequence $\text{SeqC} = C_N \dots C_1$.

[0058] Figure 6 depicts the method for implementing the method to maximize the cost of service (service charge).

10 [0059] This method is nearly identical to the iterative procedure used to maximize flow. The principle difference stems from the partitioning of clients according to their category (cost) of service: clients charged most are allocated bandwidth preferentially. This is accomplished by adding another level of iteration around the method of Figure 5. The inner iteration (steps 650 through 680) functions exactly as before, with the difference that its actions are limited to the clients belonging to

15 the given service category k (i.e. C_k). This difference also holds true of step 640 which sorts category k clients according to their flow ranges prior to entry in the bandwidth-allocating inner loop. The outer loop proceeds down a sorted sequence of service categories SeqC (generated in step 630), starting with the category generating the greatest revenue to the service provider. Given a fairly static set of

20 service categories, this sort need be performed only when the categories undergo change. Steps 670, 675 and 680 are identical to their counterparts in the method of Figure 5 (i.e. 570, 575 and 580).

-30-

[0060] The net effect of this method is preferential allocation of bandwidth according to category of service, and equitable treatment of clients within the same category of service.

An Iterative Algebraic Procedure (Figure 16)

5 [0061] The algebraic method used to maximize bandwidth (Figures 4b and 15) can be viewed as a special case of cost optimization in which all costs are equal. By construction, all sessions belonging to a given category, starting with the most expensive category, and ending with the least, and updating available bandwidth after each step in sequence, we obtain an iterative algebraic method that scales
10 much better than the preceding method.

[0062] A block diagram for this algorithm is depicted in Figure 16. Block 1600 establishes the maxima and minima for each flow. For each flow, the minimum is subtracted from the maximum to obtain the range. In block 1610, every session flow is assigned the minimum allowed value, which value will apply by default
15 should the iterative procedure run out of bandwidth before all categories can be considered. Block 1620 computes the initial value for available bandwidth, as the difference between the aggregate flow capacity and the sum of flow minima.

[0063] Flow values are obtained for each cost category in order of decreasing cost within blocks 1640 through 1675. Factor x is calculated next in block 1640, and
20 tested against 1 in block 1650. Flow rates for each sessions of the category under consideration are computed in block 1660 if x is less than 1, or 1670 in the contrary case. In the former case there is no more bandwidth to allocated to the sessions in the category under consideration.

Call Acceptance Control (CAC)CAC for Maximal Flow Overview (Figures 7-8)

5 [0064] The CAC procedure applicable to this flow optimization relies on the essential step of accepting a new client if and only if the added load induced thereby does not compromise service to existing clients or the new one. This critical step could not be accomplished without the close integration with previously- described flow-modulation methods of Figures 5, 6, 15 and 16.

10 [0065] According to the previous discussion, the minimum flow rate is the minimum sustained flow rate that guarantees that the associated viewer will not be subject to interruptions in service due to a shortfall of content from the media server. It follows that whenever data is being delivered at a rate in excess of the minimum flow rate, a downward adjustment toward the minimum level could be accommodated as needed to surrender bandwidth to any newcomer.

15 [0066] Figure 7 depicts content flow over a channel for the course of a typical session, and also how data is delivered under real-time streaming D. The amount of content delivered is the same in either case, but the manner of delivery differs considerably. A session is launched at time 0 as the network is lightly loaded, and the optimizer sets an accordingly high flow rate. Another client emerges at the end of interval 700, causing a downward adjustment to the flow rate over interval
20 B, as available bandwidth is shared between two sessions. During both of these intervals the minimum flow rate 720 drops quickly, as data accumulates in the client's media buffers. At the end of interval B, a massive influx of clients necessitates that flow be dropped to the minimum flow rate, which now lies substantially below the streaming rate D and is held until all data is delivered at the
25 end of interval C. Note that the minimum flow rate, shown as element 720, diminishes monotonically over time.

-32-

[0067] The *server swing capacity* is defined as the difference between the maximum capacity of the server and the total minimum flow rates for all active clients. Therefore:

(14) swingCapacity =

5 server.maxFlowRate - $\sum_{i \in \text{SactiveClients}} (\text{client.lookup}(i).\text{session.minFlowRate})$

[0068] Given the monotonic decreasing nature of session minimum flow rates, server swing capacity can readily be seen to be a *monotonic increasing* function of time over the intervals separating client admissions, at which point it undergoes a drop as a new load is taken on. This all-important characteristic implies the following:

Any client admitted for service based on the present value of swing capacity is guaranteed to have sufficient bandwidth at its disposal over the entire future course of the session.

[0069] Figure 8 depicts the server swing capacity 800 over the course of the sessions illustrated in Figure 7. Swing capacity rises quickly over intervals A & B as data is delivered at high flow rates over the network. It holds steady over interval C when all channels flow at their minimum rate then jumps at the end of C before resuming its monotonic rise once again.

Procedure (Figure 9)

20 [0070] In this procedure, which executes on a periodic basis, queued clients awaiting bandwidth are scanned in FIFO order. For each one, the required bandwidth is computed as per the client's prior content selection. If the available swing capacity (reduced by a safety margin) exceeds the amount required, then the client is activated and swing capacity is adjusted accordingly. Otherwise, two

25 possible cases are considered: 1) under the *FirstFit* embodiment, the procedure

-33-

continues scanning clients to the end of the queue, activating clients whose requirements can be met; 2) under the *FIFO* embodiment, the procedure ends with the first candidate client whose requirements cannot be met.

[0071] In step 900, available server swing capacity is evaluated according to the
5 formula

swingCapacity =

$$\text{server.maxFlowRate} - \sum_{i \in \text{SactiveClients}} (\text{client.get}(i).\text{session.minimumFlowRate})$$

The bandwidth requirement for client *id* in Step 920 is obtained as follows:
required_bandwidth = client.lookup(*id*).contentSelection.averagePlayRate

10 The predicate evaluated in Step 940 is given by the expression:

$$(\text{required_bandwidth} \leq \text{swingCapacity} - \text{server.cac_flowSafetyMargin})$$

[0072] In step 950, client activation entails allocation of a session and a channel,
and insertion in the set of active clients eligible for bandwidth allocation by the
15 optimal flow modulator.

[0073] In step 960 the swing capacity is diminished by the amount reserved for the
activated client:

$$\text{swingCapacity} = \text{swingCapacity} - \text{required_bandwidth};$$

Responding to Variations in Network Capacity (Maximal Flow)

20 [0074] In the CAC procedure for maximal flow, a safety margin was introduced,
namely *server.cac_flowSafetyMargin*, to provide the means for ensuring that the
server's swing capacity will never fall below a minimal threshold value.

-34-

[0075] According to this procedure, the following inequality always holds true:

(15) $\text{swingCapacity} \geq \text{server.cac_flowSafetyMargin}$

[0076] In the previous discussion, a server's swing capacity provided the basis for determining whether or not a prospective client should be allocated bandwidth.

- 5 [0077] By holding *server.cac_flowSafetyMargin* in reserve, the CAC algorithm forces delivery of content at faster than real-time rates among accepted clients, even under the heaviest load. The net effect is to apply upward pressure on client side buffer levels, thus promoting continuous accumulation of content and a jitter-free viewing experience once a session is under way.
- 10 [0078] In another embodiment, server swing capacity can also be interpreted as specifying the *maximum* amount by which the *server.maxFlowRate* constraint can be dropped without affecting service, should such an adjustment prove necessary due, for instance, to an influx of exogenous network traffic that diminishes the amount available for multi-media services. Parameter
- 15 *server.cac_flowSafetyMargin* can thus be set so as to *guarantee a minimum capacity to tighten the constraint on maximum server flow* in response to unexpected load changes that affect the server's ability to service its existing clients as well as new ones.

Anticipating Scheduled Variations in Network Capacity (Maximal Flow)

- 20 Overview (Figure 10)

[0079] Figure 10 depicts how the constraint on maximum flow might be allowed to vary according to the time of day, day of the week, and so forth, in expectation of time-varying traffic flow levels extrapolated from past experience, traffic flow models, etc. Maximum flow rate 1000 can be seen to vary based upon the time of

-35-

day. In practice, defining the right-hand-side of inequality constraint 1 as a time-dependent expression can impose such time-varying capacities. According to the previous description, the optimizer, which executes on a periodic basis, will automatically seek new flow levels for every active session as the constraint varies.

- 5 There is, however, no guarantee that an acceptable operating point will be found for all sessions (i.e. one that respects the minimal and maximum constraints on session channel flow). One such example is the case where the server is loaded to the limit and total capacity is curtailed in excess of the aforementioned safety margin. Should such a situation arise the only recourse may well be the
- 10 termination of a number of established sessions (i.e. load shedding).

- [0080] The goal is to eliminate service disruptions of this sort by allowing the CAC procedure to look ahead into the future, and accept new clients only if these can be accommodated without any compromise in service in the midst of
- previously anticipated changes* in available network capacity. The following CAC
- 15 procedure generalizes the previous one: before accepting a client, the test on swing capacity is repeated over a sequence of time segments that cover the proposed viewing period.

Definitions -

Let:

- 20 (16) $t_{\text{end}}(i) = \text{client.lookup}(i).\text{session.playTimeToGo} + t_{\text{now}}$

Let $\text{server.maxFlowRate}(t)$ be server flow capacity as a function of time, as exemplified in Figure 10.

-36-

[0081] Let $\text{Seq}_T(t_now)$ = advancing sequence of future times, lead by t_now , when $\text{server.maxFlowRate}(t)$ undergoes a step change. For instance, at 9:15 in Figure 10 this sequence reads as follows: 9:15, 9:30, 11:30, 13:30, 6:30, 7:30.

5 [0082] The server swing capacity at a future time t is computed according to the capacity and worst-case client flows at time t .

$$(17) \quad \text{swingCapacity}(t) = \text{server.maxFlowRate}(t) - \sum_{t_end(i) > t} (\text{client.lookup}(i).\text{session.minFlowRate})$$

10 [0083] It is noteworthy that the worst-case client flows at time t are expressed in terms of the present minimum flow rates, which cannot increase over time, but might hold steady. Finally, a predicate is defined that tests whether a prospective customer will cause swing capacity to be exceeded at some time t , as follows:

(18) `boolean client_fits(i ,t) {`

`if(client.lookup(i).contentSelection.averagePlayRate < =`

`swingCapacity(t) - server.cac_flowSafetyMargin)`

15 `return true;`

`else return false;`

`}`

-37-

Procedure (Figure 11)

5 **[0084]** This procedure is an adaptation of the first, which has been extended to consider swing capacity at times in the future when capacity undergoes scheduled changes. Before accepting a client, its minimal bandwidth requirement (which by construction of the flow modulator will never increase over time) is checked against *projected swing capacity* at points in time when total available capacity undergoes scheduled step changes, provided these times fall within the proposed content viewing period. A candidate is activated only if all tests succeed.

10 **[0085]** Step 1100 builds a sequence of time values (SeqT) at which step capacity changes are scheduled to occur. The first element of this sequence is *t_now*, representing the present.

[0086] Beyond step 1100 the queue of waiting clients is scanned in FIFO order, yielding a candidate designated by *id* at each iteration.

15 **[0087]** The bandwidth requirement for client *id* in Step 1120 is obtained as follows:

required_bandwidth = client.lookup(*id*).contentSelection.averagePlayRate

[0088] The worst-case end time for content flow to *id* is obtained according to the content selected, as follows:

20 *t_end* = *t_now* + client.lookup(*id*).selected.playTime

-38-

[0089] Steps 1130 through 1150 are executed within an iteration for each time point t in SeqT falling between t_now and t_end . This iteration is ended in step 1130 if t exceeds the time window of interest, or in step 1150 if the supply of scheduled capacity changes is exhausted.

- 5 **[0090]** For each time value, step 1140 compares required bandwidth to projected swing capacity.

Projected swing capacity at time t is:

$$\text{swingCapacity}(t) = \text{server.maxFlowRate}(t) - \sum_{t_end(i) > t} (\text{client.lookup}(i).\text{session.minimumFlowRate})$$

- 10 **[0091]** Note that only active clients whose t_end times occur after t are considered in the sum of minimum flow rates.

[0092] The predicate expression used in step 1140 at time t is thus:

$$(\text{required_bandwidth} \leq \text{swingCapacity}(t) - \text{server.cac_flowSafetyMargin})$$

- 15 Step 1160 performs the same actions as step 660 in the previous CAC flowchart

[0093] The first CAC process is a special case of the present one, in which the set of step change times to $\text{server.maxFlowRate}$ is empty (i.e. $\text{server.maxFlowRate}$ is constant), and $\text{Seq}_T(t_now) == t_now$.

[0094] In preparing $\text{Seq}_T(t_now)$, one need only consider future times that will pass before the longest possible content is played out if started at t_now . In order to sidestep problems associated with rollover (at midnight, year 2000, etc.), time is best expressed as a monotonically increasing value (e.g. seconds since Jan 1 1990).

CAC for Maximal Charges

Overview (Figure 13)

[0095] Previously, a method for flow modulation was presented that maximizes charges to clients with active sessions. The CAC embodiment presented previously was not sufficient as it does not consider the cost of service as a basis for connection acceptance. As a result, it may turn away higher paying customers while granting service to lower paying ones, thereby defeating the purpose for this particular embodiment. Therefore, another embodiment is defined which offers the following features:

1. Awaiting clients are serviced in order of their respective service categories, higher paying clients first.
2. Once accepted, a client is guaranteed to receive acceptable service irrespective of its service category.
3. Under heavy load conditions higher paying customers are more likely to be accepted than lower paying ones.
4. Lower paying customers will not be starved for service when higher paying ones enjoy a surplus.

-40-

5. Available bandwidth is not thrown away needlessly while clients are being denied service.

[0096] The first objective is easily met by dividing the client queue into as many bands as there are service categories, resulting in a banded queue. Bands are
5 ordered within the queue according to their service categories, with the costliest category in front. As prospective clients arrive and make their selection they are placed in their respective queue band according to their service category (which may be set contractually, according to content selection, etc.).

[0097] Our second objective is met by employing a procedure patterned after
10 those presented previously & offering the same guarantee. Toward our third and fourth objectives we propose dividing total available bandwidth in as many strata as there are service categories. As depicted in Figure 12, two service categories are shown, Premium and Basic, each entailing an associated cost of service. A prospective client is accepted only if there is sufficient swing capacity available
15 within its given service category. The swing capacity for a given category is given by the smaller of 1) the difference between its maximum floor flow rate (corresponding to the top of the stratum for the service category) and the sum of the minimum rates of all active sessions in its category or below, and 2) available swing capacity overall. Finally, our fifth objective is met by allowing the flow
20 optimizer to function freely subject to its operational constraints. The imposed ceilings on call acceptance by category relate to *minimum flow rates*, which merely impose a floor on *actual flow rates*. For example, basic clients might well consume all available bandwidth 300 in the absence of any premium customers, yet could be throttled back toward their floor flow rates (which together cannot
25 exceed 200 in this example) at any time should any premium customer suddenly demand service. In contrast, premium customers could consume the entire 300

-41-

bandwidth. As lower paying customers appear these would be admitted to the degree that their quota on minimum flow is not exceeded (i.e. 200) and the availability of swing capacity on the system.

Procedure (Figure 13)

5 [0098] The present procedure requires a number of ancillary definitions, which follow:

Let the service categories be denoted by $k = 1 \dots N$, where k also denotes the cost of service.

10 Let $\text{server.maxMinFlowRate}[k-1]$ be the top of the stratum for service category k . Note that $\text{server.maxMinFlowRate}[N-1] = \text{server.maxFlowRate}$.

Let S_k be the set of active client indices with a service category *equal to or less than* k . Note that S_1 is contained in S_2 , S_2 is contained in S_3 , and so forth, and that $S_N = S_{\text{activeClients}}$.

15 Let $\text{swingCapacity}(k)$ denote available swing capacity for service category k . By construction:

(19) $\text{swingCapacity}(k) = \text{minimum of:}$
 $(\text{server.maxMinFlowRate}[k-1] - \sum_{i \in S_k} (\text{client.lookup}(i).\text{session.minFlowRate})),$
 $(\text{server.maxFlowRate} - \sum_{i \in S_{\text{activeClients}}} (\text{client.lookup}(i).\text{session.minFlowRate}))$

20 [0099] Referring to Figure 13, this method is used for CAC when multiple rate tariffs are in effect, and there is a desire to maximize economic returns to the service provider while offering acceptable service to all.

-42-

[00100] All waiting clients are scanned in FIFO sequence. The actions taken in Steps 1320 and 1360 are identical to those described in connection with earlier CAC flowcharts.

5 [00101] Step 1340 evaluates a predicate expression that tests whether the required bandwidth can be accommodated without exceeding the lesser of 1) swing capacity available to the client's category of service, and 2) total available swing across all categories of service. The latter factor could be determinative if all available bandwidth were allocated to high paying customers, leaving lower paying ones such as the proposed client unable to draw from their unfilled quota.

10 Let us suppose that candidate client *id* belongs to rate category *k*.

We define the swing capacity available in rate category *k* as:

swingCapacity(*k*) = least of:

(server.maxMinFlowRate[*k*-1] - $\sum_{i \in S_k} (\text{client.lookup}(i).\text{session.minimumFlowRate}))$

and

15 (server.maxFlowRate - $\sum_{i \in \text{ActiveClients}} (\text{client.lookup}(i).\text{session.minimumFlowRate}))$

The predicate expression invoked by step 1340 can now be written as follows:

(required_bandwidth <= swingCapacity(*k*) - server.cac_flowSafetyMargin)

20 [00102] This algorithm processes queued clients in band sequence, and within every band in FIFO order. If the predicate evaluates to true, the client is activated. Otherwise two possible cases are considered: 1) under the *FirstFit* embodiment, the procedure continues scanning clients to the end of the banded queue, activating clients whose requirements can be met; 2) under the *FIFO* embodiment, the procedure ends with the first candidate client whose requirements cannot be met. Many other variations on these two embodiments might also be considered.

25

Anticipating Scheduled Variations in Network Capacity (Maximal Charge)

Overview

- 5 [00103] The procedure applicable to optimization of delivery charges is obtained by blending elements of the CAC method depicted in Figure 13 into the method depicted in Figure 11, which applies without change. To understand how this might work it may be useful to visualize a version of Figure 10 stratified along its length in the manner of Figure 8. As the maximum flow level undergoes a step change, so too do the widths of its constituent strata in equal proportion.

Procedure

- 10 [00104] As previously mentioned, the CAC method (Figure 11) applies to this circumstance also, provided we alter the definition of two routines, (17) and (18), upon which that procedure relies, yielding (20) and (21), and adopt the banded queue organization outlined in the previous section.

- 15 [00105] The server swing capacity at a future time t is computed according to the capacity and worst-case client flows at time t .

$$\begin{aligned}
 (20) \text{ swingCapacity}(k, t) = & \text{minimum of } (\\
 & (\text{server.maxFlowRate}(t) * (\text{server.maxMinFlowRate}[k-1] / \text{server.maxMinFlowRate}[N-1]) \\
 & \sum_{i \in S_k \text{ \& } (t_{\text{end}}(i) > t)} (\text{client.lookup}(i).\text{session.minFlowRate})), \\
 & (\text{server.maxFlowRate} - \sum_{i \in \text{SactiveClients} \text{ \& } (t_{\text{end}}(i) > t)} (\text{client.lookup}(i).\text{session.minFlowRate})))
 \end{aligned}$$

-44-

Finally, we define a predicate that tests whether a prospective customer will cause swing capacity to be exceeded at some time t , as follows:

```
(21) boolean client_fits(i ,t) {
      k = client.lookup(i).costOfService;
5      if(client.lookup(i).contentSelection.averagePlayRate < =
                                swingCapacity(k,t) - server.cac_flowSafetyMargin)
                                return true;
      else return false;
}
```

10 System Description

[00106] Referring to Figure 14, a block diagram illustrating a preferred embodiment for a system implementing the methods presented herein is shown.

[00107] Block 1485 depicts plurality of network-based client computers that receive data from a server over the course of a session. Every client sessions has
 15 an associated session block 1435 within the server by means which a flow rate of content from client to server is effected and regulated to a given flow rate set-point. The flow optimizer 1430 manages bandwidth utilization across all sessions that have been admitted by the system call admission control (CAC) and the bandwidth call admission control blocks, labeled 1400 and 1420 respectively.
 20 Specifically, the flow optimizer 1430 modulates the flow rate set-point of every active session so as to optimize aggregate flow or, more generally, cost.

Call Admission Control

[00108] Whenever a client contacts a server requesting service, the server must determine whether or not to admit the client. This admission function is
 25 performed in two separate phases. The system call admission control block 1400 considers whether or not sufficient internal resources exist to support the request.

-45-

Toward this end, the system CAC 1400 estimates the needs of the prospective client with respect to a set of limited system resources such as I/O bandwidth, cache memory, open files, threads, etc. A request is denied if any required resource cannot be obtained. In the contrary case, the request is passed on to the

5 Bandwidth CAC, which tests if the client fits and admits or rejects the client accordingly. The swing capacity is computed by the flow optimizer 1430 on a periodic basis according to equations (14) and (17) as appropriate. The managed bandwidth must reflect the true capacity of the attached network and the adapters leading thereto. This parameter should not be set arbitrarily high to prevent

10 rejection of client requests. Doing so will cause the system to admit calls that cannot be properly handled, thereby diminishing the quality of the viewing experience.

[00109] Having admitted a client, block 1430 creates a client session block 1435, which then begins to converse with its client-resident counterpart, 1480) during

15 their launch phase. Thereafter, the flow optimizer 1430 assumes control, and modulates flow over the session until either all content is delivered, or a failure to deliver is detected, or one of a number of VCR-like stream control events intervene (e.g. pause, rewind, fast-forward).

Server and Client Session Control Blocks

20 [00110] Data Transport Channel 1450 is responsible for transporting data packets originating from the flow regulator 1440 within the server to the former's peer entity 1460 within the client. The mode of communication between the peer entities may be connection-oriented (e.g. TCP, which delivers a byte stream in order, without duplication or loss) or datagram-oriented (e.g. UDP). Best-efforts

25 datagram service is typically enhanced with acknowledgements to facilitate rapid recovery from occasional packet loss, out-of-sequence delivery, and duplication.

-46-

- [00111] Peered session control channels 1445 and 1465 permit configuration and status data, together with miscellaneous commands, to be sent from client to server and vice-versa. A reliable connection-oriented transport service is typically employed for this purpose (e.g. TCP). Data and commands alike are multiplexed over this channel in the service of other entities within the clients or server. Thus, a datagram-oriented data transport channel 1460 within a client 1485 might employ the session control channel to ferry selective acknowledgements to its peer in the server. Similarly, the flow meter 1465 within the client 1485 forwards the measured flow rate to the associated flow regulator 1440 within the server.
- [00112] The flow regulator 1440 obtains content data from a cache or disk, at the current offset into the content file, which it forwards as packets to the data transfer channel 1450. The size and/or pacing of the packets are determined by the flow regulator 1440 in order to achieve the flow rate set-point imposed by the flow optimizer 1430. The flow regulator 1440 is also responsible for determining the channel flow capacity to the client, which is the least of a configured value obtained from the client and a measured value. Toward this end, during the launch phase, the flow regulator 1440 might send content data packets to the client flow meter 1475 in a series of packet trains at progressively higher peak flow rates ending with the configured maximum flow rate, if any. The highest measured flow rate reported by the client flow meter 1475 is accepted as the flow rate capacity, which is posted for use by the flow optimizer 1430. Subsequently, the flow regulator 1440 will compare the flow rate set-point with the flow rates reported by the flow meter 1465, and down grade capacity as needed should the measured flow rate fall consistently fall below the requested flow rate. Whenever such a downgrading occurs, the flow regulator 1440 will test for subsequent