香港浸會大學
HONG KONG BAPTIST UNIVERSITY

FACULTY OF SCIENCE
DEPARTMENT OF COMPUTER SCIENCE
TECHNICAL REPORT (JNG09-00) SEPTEMBER, 2000

Title:

A Transmission Scheme for Providing Streaming Support and QoS Control in a Distributed MPEG Video System[1]

Authors:

Joseph Kee-Yin, NG	(Department of Computer Science, HKBU)
Calvin Kin-Cheung Hui	(Department of Computer Science, HKBU)
Wai, WONG	(Department of Computer Science, HKBU)
Karl P.H., LEUNG	(Department of Computing and Mathematics, HKIVE – Tsing Yi)

ABSTRACT:

Recently, video transmission has made up a significant portion of the workload on today's computer networks. Because of its demands on network bandwidth, we need to have a good transmission scheme for video server, an effective buffer management, and a client/server control mechanism to manage the network delay. In this paper, we propose a new transmission scheme suitable for a video system with a multiple server design. The proposed transmission scheme is priority-driven and assigns priorities to the video data according to the urgency and the nature of the MPEG video. With the software feedback mechanism, the master server can closely monitors the system status for QoS control. According to the feedback, the system scheduler changes the feeding pattern or skips some unrecognizable frames in order to maintain the QoS that the video client has demanded. Through simulation, we show that the system can adapt to the dynamic changes in network conditions and provide the best QoS to each of its client.

Keywords:

Real-time Communications, QoS Control, Real-Time Transmission scheme, and Video Streaming

[1]The work reported in this paper was supported in part by the RGC Earmarked Research Grant under RGC/97-98/54, and by the FRG under FRG/96-97/II-103.

1 Introduction

Recently, video transmission has made up a significant portion of the workload on computer networks. Because of their demand on network bandwidth, video data need to be compressed before being transmitted over a computer network. Over the past ten years, many compression schemes have been proposed and developed for variable bit rate (VBR) video services. Of all the video compression schemes, the scheme developed by the Moving Pictures Experts Group (MPEG) is the most notable. Although MPEG [15-16] was originally designed for storing video and audio on digital media, this compression scheme is also suitable for transmitting video over a computer network. After years of development, MPEG-1 video has become one of the most popular formats for multimedia applications ranging from relatively low bandwidth applications, such as videophone and video conferencing, to higher bandwidth applications, such as, interactive video-on-demand. However, as of today's network, the bandwidth is still limited. A large amount of video data transmission often causes serious network congestion. We need to have a good transmission scheme for video server, an effective buffer management, and a client/server control mechanism to manage the network delay so that to provide a good quality of service.

A video on demand system running on top of a dedicated wide-band network is now possible (e.g. the iTV from C&W HKTelcom), nevertheless, the demand on transmitting MPEG videos over an uncontrolled network like the Internet is still growing. Because of this demand, a large amount of work has been devoted to the design of distributed video systems for streaming videos over an open network [1-5, 7-8, 19-22, 26, 28, 30]. With an uncontrollable network between the video clients and the server, three most important factors that can improve the video service are:

1. a good transmission scheme on the server side,
2. an efficient and effective buffer management on the client side, and
3. a control mechanism between the clients and the server so that the network delay, and network jitter can be taken care of.

In this paper, we propose a new transmission scheme suitable for a video system with a multiple server design. The proposed transmission scheme is priority-driven. It assigns priorities to the video data according to the urgency and the nature of the MPEG video. To make the system adaptive to the dynamic environment, we adopted a software feedback mechanism in our design with a master server closely monitoring the system status for QoS control. This master server responses to the feedback from the clients and inform the system scheduler to change the feeding pattern or skipping some unrecognizable frames in order to maintain the QoS that the clients demand.

The rest of the paper is organized as follows: Section 2 briefly surveys some related works. Section 3 describes the system architecture of our multi-server video system design. Section 4 explains the scheme for the transmission scheduling. Section 5 describes how to control the video data transmission. Section 6 explains how to manage the QoS. The next section discusses the simulation of this video server system and the resulting data. Finally, we summarize our findings and draw a conclusion.

2 Related Works

The variable bit rate (VBR) video transmission schemes and the quality of service (QoS) for streaming MPEG video over a computer network have been studied extensively. The major challenges are the problems due to the variable bit rate (VBR) characteristic, insufficient bandwidth provided for video transmission, and the uncontrollable and highly dynamic nature of the Internet traffic. To resolve these problems, a lot of work has been devoted to the design of the transmission schemes and the design of distributed video systems that consider the network congestion when streaming videos over the open network [1- 5].

For the VBR transmission schemes, most studies adopted and enhanced the traditional scheduling algorithms and use them for the real-time transmission of multimedia data [7 - 10]. Ott et al. [18] and Lam et al. [13] proposed smoothing schemes for the VBR video transmission. Pancha and El [19] studied the MPEG-1 video coding standard for transmission of VBR video and the performance of variable bandwidth allocation schemes for VBR MPEG-1 video.

Furthermore, knowing that the degradation of video quality can not be described by traditional performance measures based on frame rate and frame size, we have studied the quality of service for MPEG video in human perspective [31]. There are also extensive studies on MPEG video characterization and modeling by Ismail, Lambadaris, and Devetsikiotis, Izquierdo and Reeves, and Krunz, Sass, and Hughes [10-12].

For the quality of service support, various QoS control mechanisms have been proposed. While Bolot and Turletti proposed a rate control mechanism for transmitting packet video over the Internet [34], Reibman et al [24,35] and Reininger et al [25] made use of an adaptive congestion control scheme to study the problem of transmitting VBR video over ATM networks. Furthermore, Hasegawa and Kato had implemented and evaluated a video system with congestion control based on a two-level rate control [33]. Also in our previous studies, we suggested ways to make use of a feedback mechanism to maintain the QoS of the distributed video system [32].

On the other hand, streaming video system likes RealPlayer and RealSystem G2 from Real Network [23], VDOLive Player from VDOnet [29], InterUV Player [9], NetShow from Microsoft [17] and Streamworks from Xing Technology [27] are commercially available. However, the performance of such systems highly relies on the available bandwidth, on the utilization of a large amount of buffers. Furthermore, they do not provide much QoS guarantees on the video, not to mention that each of these systems has its own proprietary video format.

In general, there are two different approaches in developing video streaming systems. The first is to develop coding algorithms that are suitable for transmitting video data through the network. These algorithms often use error concealment techniques to reduce the retransmission of packets. Another approach is to develop new

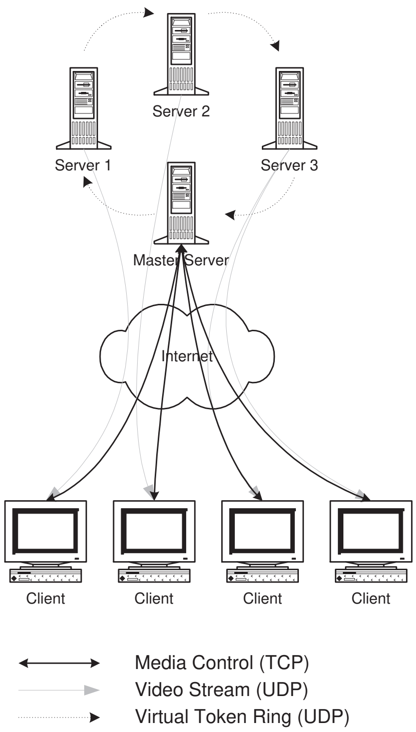

Figure 1: The Overall System Architecture of the Distributed Video System.

protocols, introduce new mechanisms such as Realtime Transport Protocol and Realtime Streaming Reservation Protocol,$\cdots$ etc [14] in order to support the real-time services. In our case, we adopt none of the above in our video system. Since MPEG is an open standard for video applications, we do not want to develop yet another coding scheme for the video transmission or for MPEG video pre-processing. Furthermore, we do not invent or develop special transport protocols for the video transmission. Instead, we will impose a control mechanism and make use of the client's buffer and the server's transmission scheme to take care of the network delay and network jitter.

In summary, there are relatively few video systems that can provide the support for streaming MPEG video over an uncontrolled network while still providing some guarantee on the Quality of Services (QoS) of the video being displayed. This is what our system can do, and can do better.

3 The System Architecture

Our design of a Video on Demand (VoD) system is distributed, and it uses a multi-server architecture. Figure 1 shows the overall system architecture of our design. It consists of a master server, a set of video servers and a number of clients. The master server controls the operation of the entire system. The video servers are responsible for sending video data to the clients. The clients are for displaying the video. They also provide feedback to the servers to indicate their operating status.

The master server communicates with the video servers via a token-passing protocol similar to the token-bus MAC protocol. The communication between the servers and the clients is through the Internet that is uncontrollable by our system.

3.1 The Master Server

The functions of the master server are:

1. To accept requests from the clients,
2. To allocate video servers to serve the request,
3. To monitor the overall system status, and regular the flow of the video data so that the QoS is maintained.

A video request is initiated by a client. It sends the request to the master. After receiving the request, the master will select the least loaded video server to service the request. A connection between the video server and the client is established. While the video server sending the video data to the client, the master server acts as a scheduler to regulate the flow of the data according to the network conditions. The main purpose of the scheduling is to maintain as good a QoS as the network condition allows. The regulation is achieved by varying the amount of time during which the video server can transmit data. The master server communicates with the video servers by passing tokens, in a fashion similar to the token bus MAC protocol. The scheduling algorithm will be described in Section 4.

In order to maintain the schedule of all the video servers, the master server provides a synchronization service. It keeps a system timer for the synchronization. The design and implementation for the clock synchronization and token maintenance are outside the scope of this paper.

3.2 The Video Servers

The video servers are responsible for sending video data to the clients. They operate under the control of the master server. This control is realized by passing tokens. When a video server receives a token, it invokes its own local scheduling function to regulate the data transmission of the video streams it serves. At all time, the data transmission is governed by the QoS parameters.

3.3 The Clients

The major function for the clients is to receive the video frames from the video server and to display the frames on time. Each client has its own video buffer to store the received data. When the client displays a frame, if there is a status change, it sends a feedback to master server. The information sent back to the server includes the client's status, the QoS it requires and the buffer status.

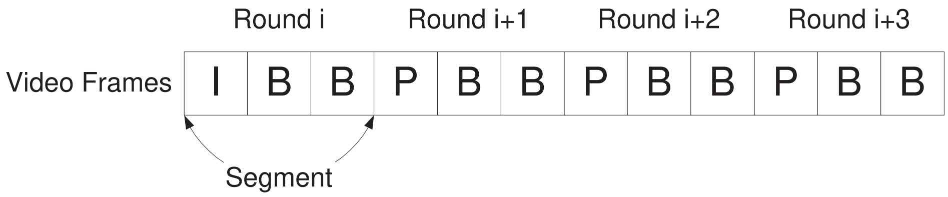

Figure 2: A GOP of MPEG video to be transmitted in 4 rounds.

Transmission	Real-time		Non real-time	
Session	SESSION 0	SESSION 1	SESSION 2	SESSION 3
Description	I or P-frame only. Drop the frames when the cycle reach the limit.	B-frame only. Drop the frames when the cycle reach the limit.	Transmit one segment for buffering.	For advance buffering under transmission control.

Table 1: Summary of transmission scheme in 4 sessions

4 Transmission scheduling scheme

The service discipline of this scheduling scheme divides the video transmission into rounds. Each round is a unit for transmission management. It contains a number of video frames. A round is defined by a period of Tround in msec. For example, if we define a round to be 100 msec and the frame rate of the video is 30 frames per second, then for each video stream during a round, three frames have to be transmitted. In the case that the server is serving more that one client, all streams must complete the transmission of three frames within the period.

Since a MPEG video clip is structured as a sequence of groups of pictures, it is desirable to align the rounds to the beginning of a group of pictures. Therefore, we can conceptually divide the video sequence into segments. Each segment contains all the frames to be transmitted in a round. For instance, Figure 2 shows a GOP being divided into four segments and being transmitted in four rounds. Each segment contains either an I-frame or a P-frame, plus two B-frames.

The frames transmitted within a round are classified into two categories: real-time frames and non-real-time frames. The real-time frames are those that are required to be transmitted within each round. Thus, the number of real-time frames is equal to the number of frames in a segment. The non-real-time frames are those that are transmitted with no time requirement, such as the frames for initial buffering and advance buffering.

We divide each round into four sessions, namely, SESSION 0 to SESSION 3 with SESSION 0 bearing the highest priority. Hence, the master server starts each round with SESSION 0. It switches to the next session when the current session is completed. In each session, every video servers takes turn for video transmission. The first two sessions, SESSION 0 and SESSION 1, are designed for the transmission of real-time frames.

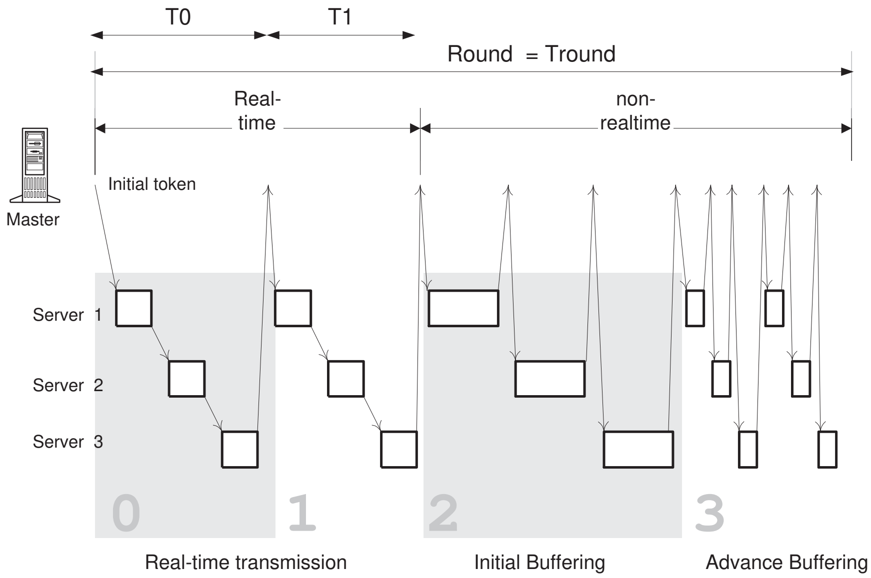

Figure 3: Sessions sequence.

Furthermore, SESSION 0 is for transmitting only I-frame or P-frame while SESSION 1 is for B-frames. The last two sessions, SESSION 2 and SESSION 3, are used to transmit non-real-time frames. The frames for initial buffering are transmitting in SESSION 2 where it is only allowed to transmit one segment per round. The last session is for advance buffering. It makes use of the remaining time in each round to transmit extra video frames. However, the number of extra frames transmitted in SESSION 3 is limited and the detail would be discussed in transmission control section. Table 1 summarizes the function of the sessions.

In any time, video transmission is under QoS control. This means video frames may be dropped by the QoS control when the bandwidth demand exceed the current bandwidth available. The details on how to drop the bandwidth demand with minimal video degradation will be discussed in latter sections.

Each round must complete within the period T_{round}. At the end of a round, the current session is stopped. The next round starts with SESSION 0. If the transmission cycle reaches the end of a round while still in SESSION 0 or 1, the unsent frames are dropped.

At the beginning of each session, the master server initiates a token. The token is passed to all video servers in turn in SESSION 0 and SESSION 1. It is returned to the master server at the end of these sessions. Then, the master server initiates a token for the next session. The video server who possesses the token will transmit the video frames. In SESSION 2 and SESSION 3, the token is passed in a different way. The master server passes the token to a video server. It is returned to the master as soon as the video server completed its session. It will not be passed to the next video server.

Figure 3 shows an example for a 3-video server system. At the beginning, the master server initiates a token and sends it to Server 1. After Server 1 completes the I-frame or P-frame transmission, it passes the

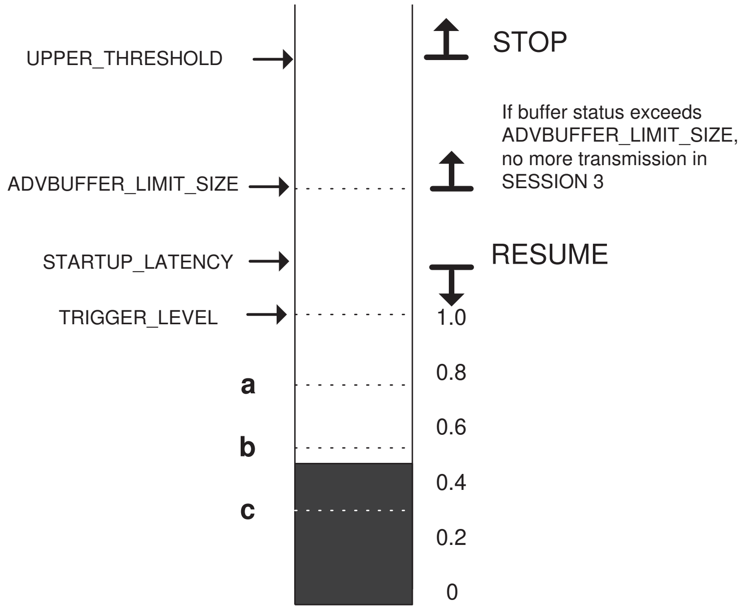

Figure 4: Transmission control mechanism.

token to Server 2, and so on and finally the token is returned to the master server. This completes SESSION 0. SESSION 1 works in a similar way. Groups of B-frames are transmitted by the video servers. For non-real-time transmission sessions, the master server passes the token to the video server individually. It also specifies the duration each video server can hold the token. Thus, it keeps the transmission under a tight control.

5 Transmission Control

The transmission control mechanism is based on a "Stop and Resume" control mechanism. The basic idea is to monitor the buffer status. Referring to Figure 4, when the buffer exceeds the upper threshold, defined as UPPER_THRESHOLD in terms of bytes, the client sends a stop command to the server to stop the transmission. Meanwhile, the buffer is being continuously consumed, and it eventually gets below the upper threshold. The transmission will resume when the buffer drops below the STARTUP_LATENCY.

The disadvantage of the "Stop and Resume" mechanism is the fluctuation of buffer. This cycle of stopping and resuming makes the buffer status fluctuates in large amplitude. In order to achieve a more stable buffer status, we need to control the server's transmission rate. Our scheme bounds the transmission rate by limiting the number of frames being transmitted in SESSION 0 to SESSION 2. In SESSION 3, two parameters control the transmission rate. The first, ADVBUFFER_LIMIT_SIZE, limits the maximum number of frames allowed to transmit for each video stream. The second is the status of client's buffer. If the buffer status exceeds the ADVBUFFER_LIMIT_SIZE, that stream should not be transmitting any frames in SESSION 3.

Transmitted Frame(s)	Skipped Frame(s)	GoP Pattern
12	0	IBBPBBPBBPBB
11	1	IBBPB.PBBPBB
10	2	IBBPB.PBBPB.
9	3	IBBPB.PB.PB.
8	4	IB.PB.PB.PB.
7	5	IB.PB.P..PB.
6	6	IB.P..PB.P..
5	7	I..P..P..PB.
4	8	I..P..P..P..
3	9	I..P..P.....
2	10	I..P........
1	11	I...........

Table 2: GoP Pattern used for Dropping and Adding of Frames.

6 Quality of Service Control

The measurement of video quality of our video system is based on QoS Control in human perspective (QoS-Human) as investigated in our previous study [31]. QoS-Human can truly reflect the true video quality viewed by human instead of the traditional metrics, based on frame rate (QoS-Frame) and by frame size (QoS-Byte). When the current available bandwidth is not enough to transmit all the scheduled frames in SESSION 0 and 1, we have to drop frames according to QoS-human. This reduces the bandwidth demand and keeps video degradation to a minimum. We assume that the group of pictures of the MPEG video has the pattern IBBPBBPBBPBB. The representative patterns when various number of frames are skipped is shown below in Table 2.

6.1 Overview of QoS control mechanism

In order to make effective use of the network bandwidth and to avoid over-loading the network, QoS-control automatically drops the frames from each active video to reduce the network bandwidth. The QoS control mechanism is implemented in the clients and in the master server. The QoS control on the client side is called the client QoS. It is determined according to the client's buffer status. The QoS control in the video servers is called the server QoS. It is determined by the scheduler in the master server. The QoS controller in the master server receives both server QoS and client QoS. It chooses the lowest QoS as the final QoS for QoS control. This QoS is used to control the dynamic QoS filter in each of the video servers. Figure 5 below illustrates the flow of the QoS Control.

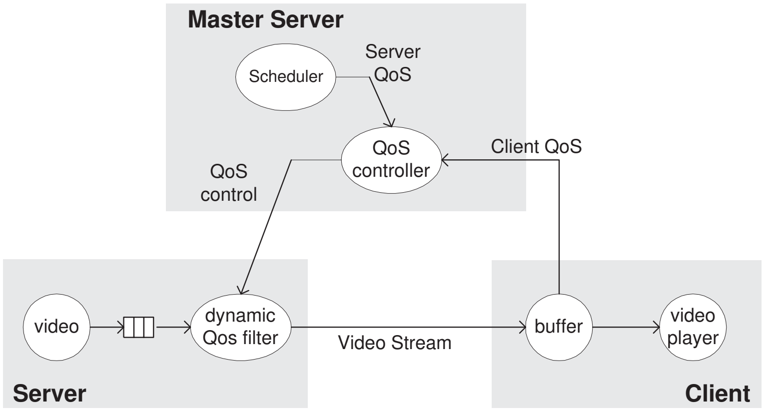

Figure 5: Overview of QoS Control.

Transmitted Frame(s)	Skipped Frame(s)	GoP Pattern	Condition
12	0	`IBBPBBPBBPBB`	buffer > TRIGGER_LEVEL
10	2	`IBBPB.PBBPB.`	buffer > a
8	4	`IB.PB.PB.PB.`	buffer > b
6	6	`IB.P..PB.P..`	buffer > c
4	8	`I..P..P..P..`	buffer $\leq$ c

Table 3: Selected GoP pattern for Dropping and Adding Frames.

6.2 Client QoS controller

As mentioned in the previous sections, we have set the STARTUP_LATENCY in the buffer such that there will be pre-buffered frames before starting to display the MPEG video. Because of the VBR nature of the MPEG video, this pre-buffering is necessary, otherwise, the client may experience buffer underflow.

With this, we want to maintain the same buffer level at all time during the video display. However, due to network congestion and video servers' transmission, the buffer level might drop below the STARTUP_LATENCY. Hence, we set up a TRIGGER_LEVEL together with three more parameters to control the client QoS requirement.

Basically, the client QoS dynamically changes according to client buffer status. We define a value TRIGGER_LEVEL to mark the point where frame dropping may be triggered. If the buffer status is higher than the TRIGGER_LEVEL, then the network bandwidth should be enough for the current QoS requirement. No frame dropping is needed. When network is congested, the buffer level will drop. When it drops below the TRIGGER_LEVEL, the client will drop the QoS requirement and demand the video server to drop some frames in order to reduce the network load. The lower the buffer, the more the frames are skipped at the video server and hopefully the faster the recovery of the network.

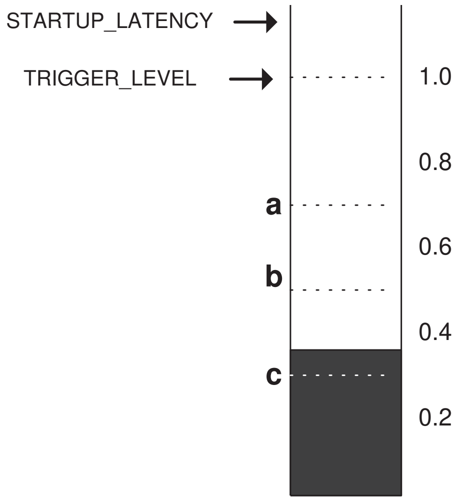

Figure 6: Client QoS control mechanism.

Table 3 and Figure 6 show how the TRIGGER_LEVEL and its parameters a, b, and c are set in the buffer and how to determine the number of frames needed to be dropped for the QoS Control. Note that the frame skipping pattern is not arbitrary, it is based on the results of our QoS study in human perspective [31].

6.3 Server QoS controller

The server QoS is to detect whether the current bandwidth can support the transmission demand of the bandwidth. Referring to Figure 7, let T0, T1 be the time needed for the video transmission in SESSION 0 and SEESION 1, respectively.

If T0 > T_{round}, the system is unschedulable even under the minimum QoS requirement. If T0 + T1 > T_{round}, the system is unschedulable under current QoS requirement. Hence, it needs to lower the QoS for a number of streams. To drop the Server QoS, we find the stream with the highest QoS among all active video streams, and decrease its QoS.

On the other hand, if T0 + T1 < T_{round}, the current network bandwidth is sufficient for the current QoS request. The master server can increase the server QoS. It chooses the video stream having the lowest QoS among all active video streams, and increases its QoS.

7 Simulation Experiments and Results

7.1 Simulation Setup

In order to evaluate the proposed scheme, we have written a program to simulate multiple video servers with multiple clients connecting through an open network. The simulation program is executed on the Intel Platform

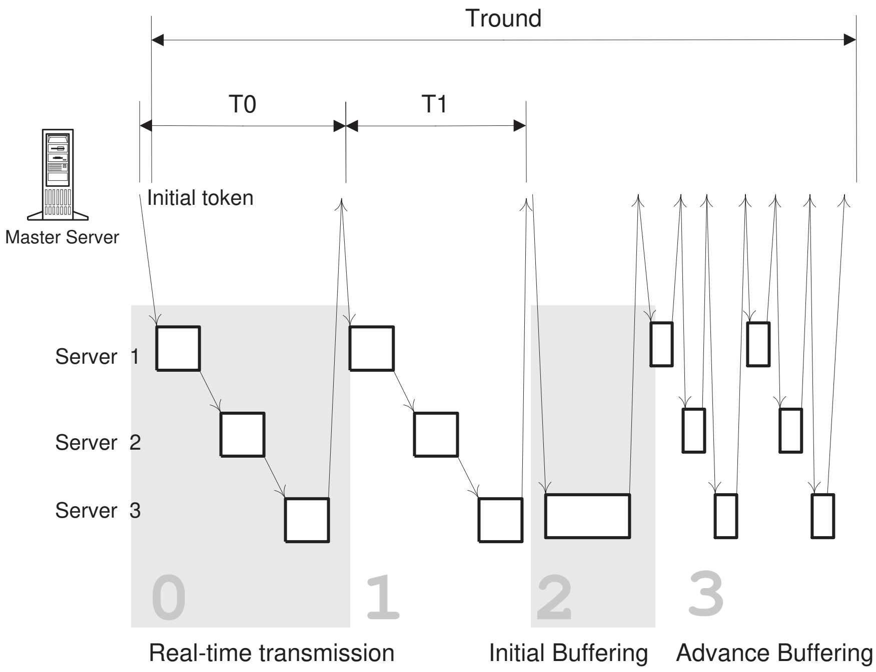

Figure 7: Client QoS control mechanism.

(Pentium II-300) with the following assumptions and system parameters:

- All video clients start requesting the server for a MPEG video stream at the same time. This simulates the worst case phasing of the video transmission.
- Each video server services multiple streams and sends the video data to the clients a packet at a time.
- Each packet has a fixed size of 2048 bytes.
- Packets are delayed by the network. The delay time used in the simulation is the real delay time between two Internet hosts located on two different continents. The data are collected using packets of the same size as in the simulation.
- We assume a 10 Mbps bandwidth limit on the open network.
- The simulation time is 180 seconds.
- An action video is used.
- The number of clients being used varies in the range of 3 to 8. The network bandwidth is sufficient for 3 steams with no skipped frames.
- The upper threshold of the client buffer (in MByte) is set to UPPER_THRESHOLD = 0.8
- The startup latency (in MByte) is set to STARTUP_LATENCY = 0.35

- The trigger level(in MByte) is set according to the formula below:

  1. TRIGGER_LEVEL = STARTUP_LATENCY - (maximum I-frame + 2 * maximum B-frame)
  2. The Advance buffering level(in MByte) is equal to STARTUP_LATENCY

7.2 Simulation Results and Discussion

We have the following criteria for the performance measures:

- *Number of effective frames stored in the buffer*: This is the number of frames including the frames skipped by QoS controller. We named this virtual number of frame (vnof). This is proportional to the time that is available for the client to display. The rate of vnof decrease reflects the low transmission rate due to network congestion.
- *The Quality of service the client perceived*: The higher the QoS perceives, the more the number of frames displayed by the client.
- *Buffer status*: It is the video frames stored in client's buffer. We always related this to the transmission rate for the video stream.

We first discuss the overall performance for the scheduling scheme. Then, we discuss the system parameters individually.

7.2.1 Overall performance

The parameters used are as follow:

- Trigger level for client QoS are $a = 0.4$, $b = 0.3$, $c = 0.2$
- Buffering limits are ADVBUFFER_LIMIT_SIZE = STARTUP_LATANCY and ADVBUFFER_LIMIT = 1

The simulation results are shown in Figures 8, 9 and 10. we have the following observations:

- For all cases, the client begins to display video when it has buffered about 60 frames. The buffer rate increases initially. When reaching the buffering limit, it begins to stabilize. This means the rate of input and output is under control.
- For 3 and 6 video streams, the final vnof for all streams settles at the same level at 80 frames as shown in Figure 8 and Figure 9. In Figure 10, the vnof of 8 streams settles at different levels ranging from 70 to 110. This indicates that the network bandwidth is insufficient for supporting all 8 streams.

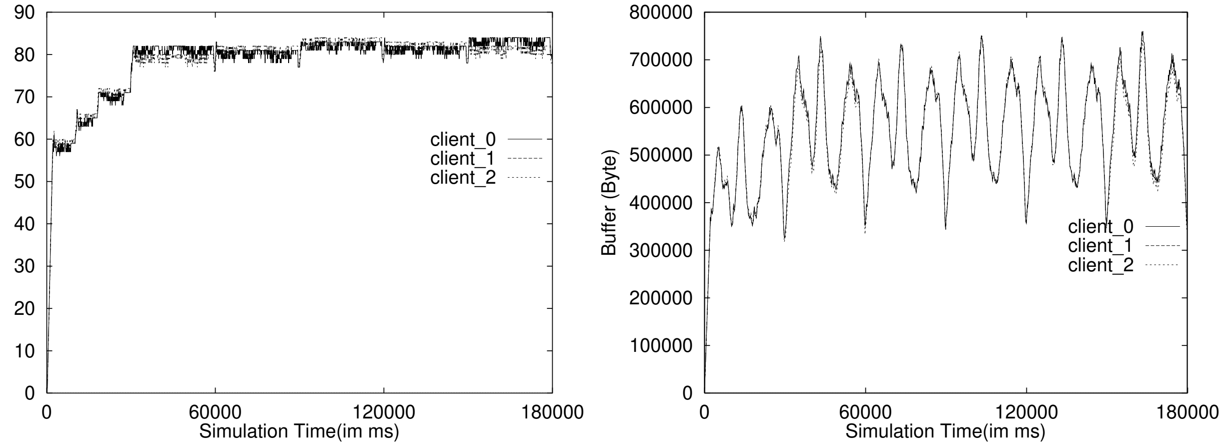

Figure 8: Overall performance for 3 streams

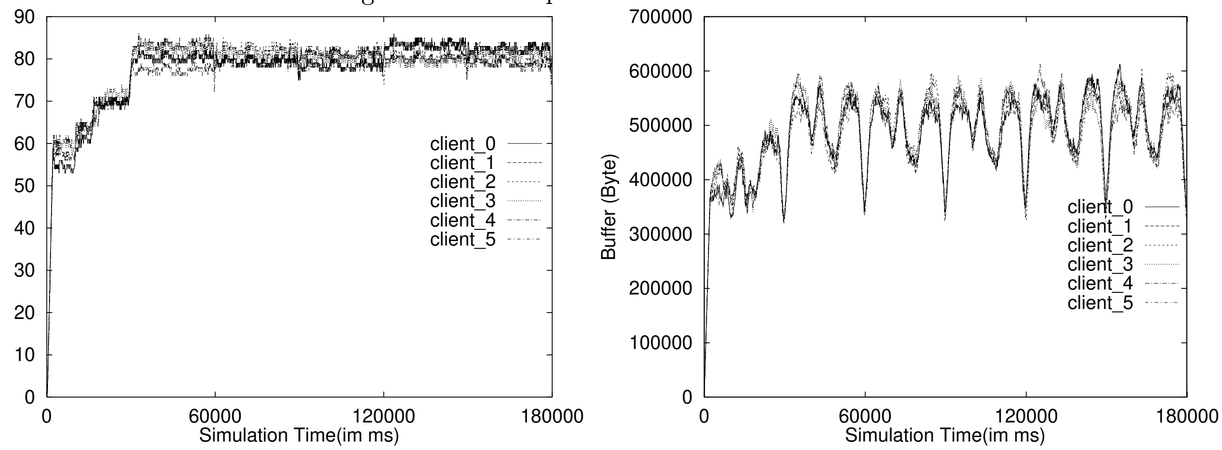

Figure 9: Overall performance for 6 streams

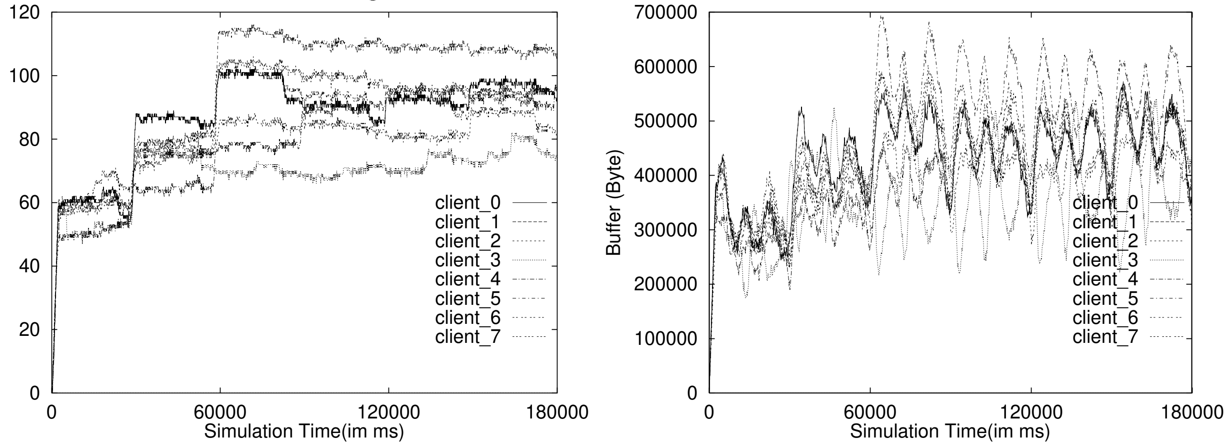

Figure 10: Overall performance for 8 streams

No. of stream	Average QoS	s.d. QoS	Max. Buffer (Byte)
2	11.9920	0.001414	756852
3	11.9927	0.002887	766061
4	11.9895	0.001915	769662
5	11.6008	0.065067	720706
6	10.1550	0.022262	616839
7	8.3687	0.028447	577594
8	6.6990	0.066214	703246
9	5.1473	0.089638	902325

Table 4: Overall Result of the Scheduling Scheme.

- The Average Quailty-of-Serivce for 3 streams drops slightly caused by the QoS control. Since a variable bit rate video is being transmitted. Some portions of video that have a large bit rate cause the QoS controller to drop the QoS (as shown in the first row of Table 4).
- The buffers are kept under control even when 8 streams are sharing the network bandwidth, but the average QoS drops to 6.6990 in order to maintain a fair transmission among the video streams.
- The maximum buffer status for all cases has not exceeded the buffer maximum bound. This is well controlled by the transmission rate control.

7.2.2 Selection of Client QoS parameters

The client QoS is controlled by the feedback mechanism. After the start of displaying video, the client feeds back the buffer status to the master server. According to buffer consumption, the new value of the Client QoS will be determined so as to lower the QoS requirement when the network is congested and raise the QoS requirement when the network can provide the support.

Since the change of the client QoS is triggered by the setting of the TRIGGER_LEVEL, we want to investigate the effect of different settings, as well as its related parameters, namely, a, b, and c in this experiment.

The parameters a, b, and c are normalized to the TRIGGER_LEVEL with the trigger level set at the same as the STARTUP_LATENCY. We conduct the experiments using 8 video streams and disable the Advance Buffering, i.e. ADV_BUFFER = 0. We have tried different sets of the QoS parameters and the results are shown in Table 5.

We observe that Set 1 produces the best performance, i.e., Set 1 has the highest Average QoS and the smallest value in standard deviation. We also observe that Set 5 through 7 produced a low Average QoS and a large standard deviation. Hence, in general, there is a good indication that the setting of a, b, and c should be in the lower half of the TRIGGER_VALUE in order to have a good performance results.

Test Set	Client QoS Tirgger			Average QoS	s.d. QoS
	a	b	c		
1	0.4	0.3	0.2	7.1248	0.1319
2	0.6	0.4	0.2	7.0938	0.3207
3	0.7	0.4	0.1	6.9395	0.4479
4	0.85	0.3	0.1	6.8610	0.2209
5	0.7	0.6	0.4	6.7089	0.9279
6	0.8	0.6	0.2	6.6384	0.9279
7	0.8	0.6	0.4	6.5113	0.7858

Table 5: Result of Various Client QoS Parameters.

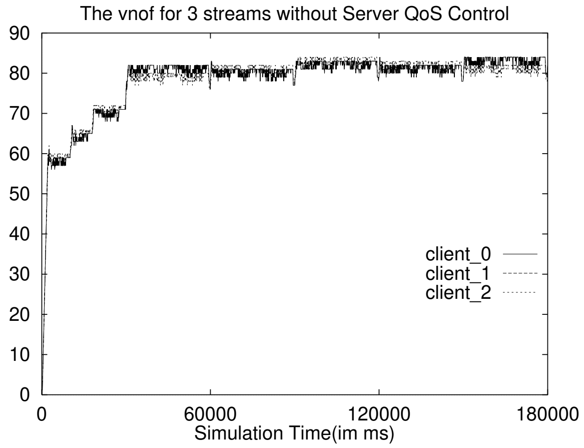

Figure 11: The vnof for 3 streams without Server QoS control

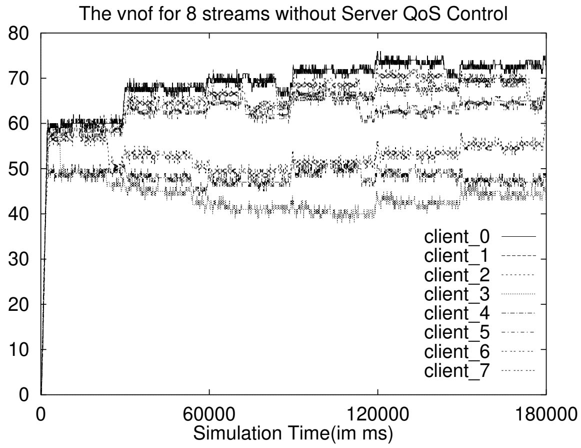

Figure 12: The vnof for 8 streams without Server QoS control

7.2.3 Effect on Server QoS control

The Server QoS control takes effect when there is insufficient bandwidth. When the scheduler detects that, the streams are unschedulable under the current QoS. The QoS controller tries to lower the video quality by skipping frames until the system is schedulable. If the bandwidth is sufficient, the Server QoS control has no effect. It shows similar result for the 3-stream case by comparing Figure 8 with Figure 11 and Table 4 with Table 6.

However, for the 8-stream case, the bandwidth is not enough to support the current QoS demand. Hence, some streams suffer by being assigned a lower QoS. If without any server QoS control, the drops in QoS are not the same across the 8-streams. Figure 12 shows that some streams are having a vnof of 35 while other streams are still having a vnof of 75. By comparing Figure 10 with Figure 12, we can see that the variance of the Average QoS among the 8 streams is much larger than the system that adopts the server QoS control.

No. of stream	Average QoS	s.d. QoS	Max. Buffer (Byte)
2	11.9920	0.001414	756852
3	11.9927	0.002887	766061
4	11.9895	0.001915	769662
5	11.6576	0.019269	712006
6	10.5013	0.040844	602573
7	8.9986	0.053820	530361
8	7.16975	0.364730	464487
9	6.0509	1.165740	889483

Table 6: Result of scheduling scheme without Server QoS control.

Although, in the 8 streams case, the system without QoS control has a higher Average QoS (7.1698) than the system that adopts the QoS control (6.699), its variance among the 8 video streams is much larger (0.365 comparing to 0.066). Hence, our proposed QoS control is much fairer to every video stream.

7.2.4 Effect on different settings of ADVBUFFER_LIMIT

According to our proposed transmission scheme, SESSION 0 and SESSION 1 are for transmitting video frames with real-time constraints. We transmit the I- or P-frames first in SESSION 0, and transmit the B-frames in Session 1. After the above mentioned real-time transfer, the remaining time in a round will be utilized for non-real-time transmission. While SESSION 2 is for the frames for initial buffering or startup latency, SESSION 3 is for the advance buffering.

In general, we want to have advance buffering so that for a sudden congestion on the network, the video will not be interrupted. However, we also want to keep the advance buffer low for conserving buffer space. Hence, for this experiment, we focus on the effect of different settings of the ADVBUFFER_LIMIT. To transmit frames in SESSION 3, it needs to satisfy the following conditions:

- ADVBUFFER_LIMIT_SIZE = 2 * UPPER_THRESHOLD
- ADVBUFFER_LIMIT uses one of the following values: 0, 1, 16

Setting ADVBUFFER_LIMIT to 0 means that we do not allow any buffering in SESSION 3. Setting ADVBUFFER_LIMIT to 1 means that each stream is allowed to send only one frame for advance buffering in each round. Setting ADVBUFFER_LIMIT to 16 means that, for each stream, up to 16 frames can be sent for each cycle if time permits. For normal circumstances, there is little time left in SESSION 3, hence, each stream can only send 1 to 2 frames during SESSION 3.

The followings are some observations from this simulation:

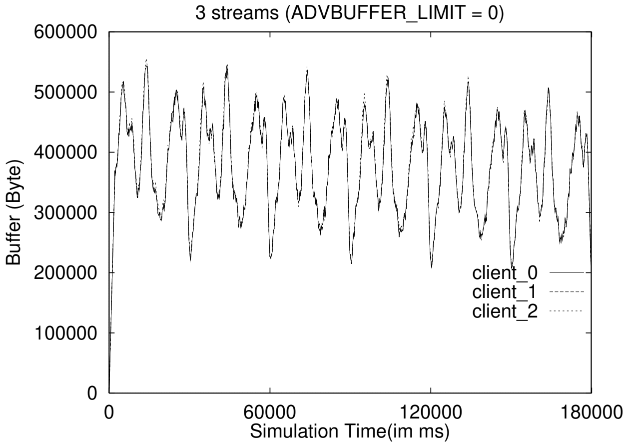

Figure 13: 3 streams and ADVBUFFER_LIMIT = 0

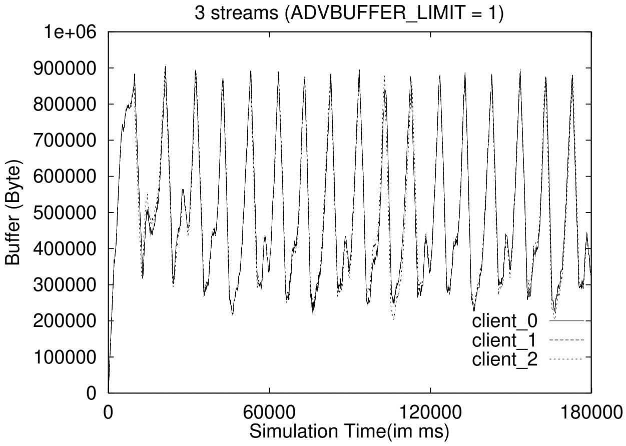

Figure 14: 3 streams and ADVBUFFER_LIMIT = 1

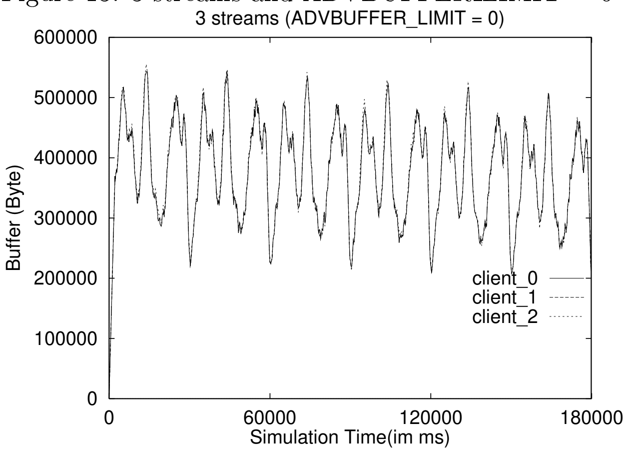

Figure 15: 3 streams and ADVBUFFER_LIMIT = 16

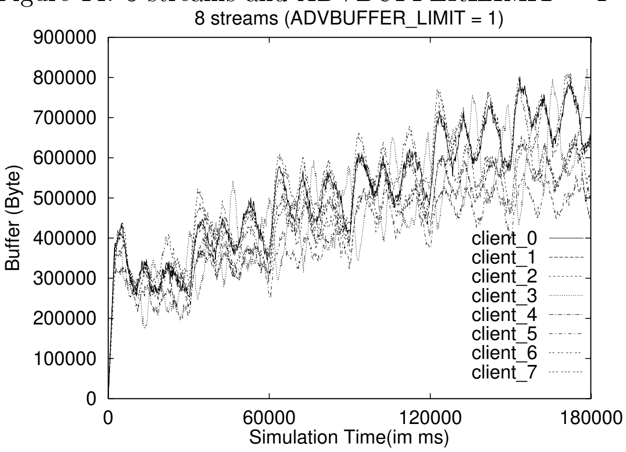

Figure 16: 3 streams and ADVBUFFER_LIMIT = 0

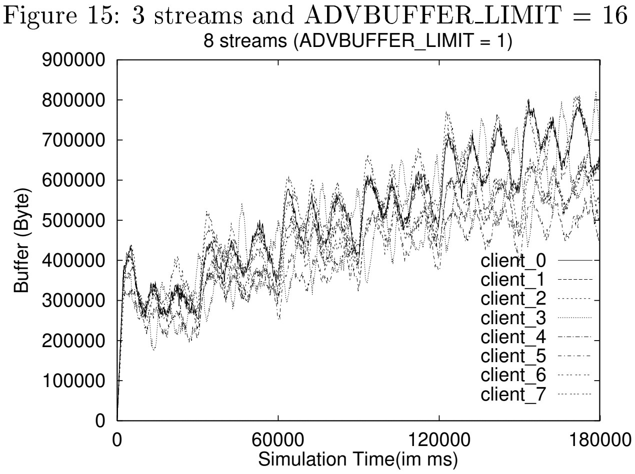

Figure 17: 3 streams and ADVBUFFER_LIMIT = 1

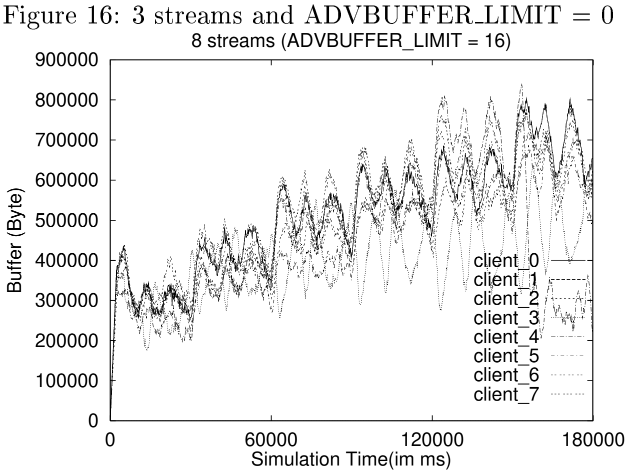

Figure 18: 3 streams and ADVBUFFER_LIMIT = 16

No. of streams	ADVBUFFER_LIMIT	Average QoS	s.d. QoS	Max. Buffer (Byte)
3	0	11.3240	0.0763	569616
3	1	11.3853	0.0023	939345
3	16	11.7077	0.0204	994131
8	0	6.7143	0.0801	445265
8	1	6.7529	0.0430	822439
8	16	6.7490	0.0522	854599

Table 7: ADVBUFFER_LIMIT vs Performance measure.

- Without transmission control bounded by limiting the buffer size, the buffer status became fluctuated. It causes the buffer to exceed the upper threshold.
- ADVBUFFER_LIMIT does not directly affect the Quality-of-Service, but it affects the size of the buffer. The maximum size of the buffer is large for while ADVBUFFER_LIMIT is greater than 1.
- If there are only 3 streams, the network bandwidth is large enough for buffering. Figure 13 shows the normal buffer status. If the buffer limit is set to higher value, the transmission rate will increase as shown in Figure 14 and Figure 15. At the same time, the fluctuation of the buffer is larger.
- If there are 8 streams, the network bandwidth is insufficient for all streams. As Figure 16 shown, the buffer consumption will increase due to low transmission rate.
- The buffer status is similar for buffer limit being 1 and 16 as shown in Figure 17 and Figure 18. It is because the network bandwidth was fully utilized. It is impossible to increase the transmission rate.

8 Conclusion

In summary, there are relatively few video systems that can provide the support for streaming MPEG video over an uncontrolled network while still providing some guarantee on the Quality of Services (QoS) of the video being displayed. In this paper, we propose a new transmission scheme suitable for a video system with a multiple server design. The proposed transmission scheme is priority-driven. It assigns priorities to the video data according to the urgency and the nature of the MPEG video. To make the system adaptive to the dynamic environment, we adopted a software feedback mechanism in our design with a master server closely monitoring the system status for QoS control. This master server responses to the feedback from the clients and inform the system scheduler to change the feeding pattern or skipping some unrecognizable frames in order to maintain the QoS that the clients demand. And finally, our simulation study has shown that the proposed distributed video system can adapt to the dynamic changes in network condition and provide the best QoS to each of its clients.

References

[1] Veronica Baiceanu, Crispin Cowan, Dylan McNamme, Calton Pu and Jonathan Walpole, "Multiple Applications Require Adaptive CPU Scheduling", In *Proc. of Workshop in Multimedia Resource Management*, December 1-2, 1996.

[2] J. Bolot and T. Turletti. "A Rate Control Mechanism for Packet Video In the Internet". in *Proc. of INFOCOM'94*, pp 1216-1223, 1994.

[3] W. Bolosky, J. Barrera III, R.Draves, R. fitzgerald, G. Gibson, M.Jones, S. Levi, N. Myhrvold, and R. Rashid, "The Tiger Video Fileserver", In *Proc. The 6th International Workshop on Network and Operating System Support for Digital Audio and Video*, April 1996. Also available at http://www.research.microsoft.com/ in the operation systems area.

[4] W. Bolosky, R. Fitzgerald, and J. Douceru, "Distributed Schedule Management in the Tiger Video Fileserver", In *Proc. SOSP'97*, Also available at http://www.research.microsoft.com/rearch/os/bolosky/sosp/cdrom.html.

[5] J. F. Koegel Buford, *Multimedia Systems*, Addison Wesley, 1994.

[6] Shanwei Cen, Calton Pu and Richard Staehli, " A Distributed Real-time MPEG video Audio Player", in *Proc. of the 5th International Workshop on NOSSDAV'95*, April 18-21, 1995.

[7] Cripsin Cowan, Shanwei Cen, Jonathan Walpole and Calton Pu, "Adaptive Methods for Distrubuted Video Presentation", *ACM Computing Sureys*, voulme 27, number 4, pp.580-583.

[8] D. Le Gall, "MPEG: A Video Compression Standard for Multimedia Applications". *Communications of the ACM*, volume 34, nubmer 4. pp. 46-58, 1991

[9] Steven Gringeri, Bhumip Khasnabish, Arinanne Lewis, Khaled Shuib, Roman Egorov, and Bert Basch, "Transmission of MPEG-2 Video Streams over ATM", *IEEE Mutlimedia*, January-March 1998, pp 58-71.

[10] T. Hasegawa, T. Hasegawa, T. Kato "Implementation and Evaluation of Video Transfer System over Internet with congestion control based on Two Level Rate Control", in *Proc of RTSCA'99*, pp.141-148, 1999

[11] InterVU from InterVU (www.intervu.com)

[12] M. Ismail, I Lambadaris, and Devetsikiotis, "Modeling Prioritized MPEG Video Using TES and Frame Spreading Strategy for Transmission in ATM Networks", *Proc. IEEE INFOCOM*, April 1995.

[13] R. Izquierdo, and D. Reeves, "Statistical characterization of MPEG VBR video at the SLICE layer", *Proc. SPIE Multimedia computing and Networking*, Vol. 2417, Spetember 1995.

[14] H. Kanakia, P Mishra and A. Reibman. "An adaptive Congestion control Scheme for Real-Time Packet Video Transport" *Proc. of ACM SIGCOMM '93*, pp20-31, Sep. 1993.

[15] M. Krunz, R. Sass, and H. Hughes, "Statistical Characteristics and Multiplexing of MPEG streams", *Proc. IEEE INFOCOM*, April 1995.

[16] S. Lam, S. Chow, and D. Yau, "In Algorithm for Lossless Smoothing of MPEG Video", *Proc. of ACM SIGCOMM 94*, pp.281-293, London England UK, 1994.

[17] Chunlei Liu, Multimedia Over IP: RSVP, RTP, RTCP, RTSP (Available at http://www.mpeg.org)

[18] Joan L. Mitchell, William B. Pennebaker, Chad E. Fogg, and Didier J. Legall, *MPEG Video Compression Standard*, pp. 1-49.

[19] MPEG starting Points and FAQs, (Available at http://www.mpeg.org/MPEG/starting-points.html)

[20] NetShow from Microsoft (Available at http://www.microsoft.com)

[21] Joseph Ng, Karl Leung, and Wai Wong, "Quality of Service for MPEG Video in Human Perspective", *Technical Report*, Department of Computer Science, Hong Kong Baptist University, July 2000. (`http://www.comp.hkbu.edu.hk/ jng/Tech-Rpt/JNG07-00.ps`)

[22] T. Ott, T. Lakshman, and A. Tabatabai, "Scheme for Smoothing Delay-sensitive Traffic Offered to ATM Networks", *Proc. of IEEE INFOCOM'92*, pp.776-765, 1992.

[23] Pramond Pancha and Magda El Zarki, "Bandwidth Requirement of Variable Bit Rate MPEG sources in ATM Netowkrs" , in *Proceeding of IEEE INFOCOM'93*, pp902-909, 1993.

[24] K. Patel, B. C. Smith, and L. A. Rowe "Performance of a Software MPEG Video Decoder", *Proceedings of ACM Multimedia 93*, pp 75-82

[25] Calton Pu and R. Fuhere, "Feedback-Based Scheduling: a Toolbox Approach", in *Proceeding of 4th Workshop on Workstation Operation Systems*, October 14-15, 1993.

[26] RealPlayer and RealSystems G2 form RealNetworks (www.realaudio.com)

[27] R. Reibman, and A. Berger, "On VBR video Teleconferencing over ATM Networks" *Proc. of IEEE GLOBECOM'92*, pp. 314-319, 1992.

[28] R. Reininger, D. Raychaudhuri, B. Melamed, B. Sengupta, and J.Hill, "Statistical Multiplexing of VBR MPEG Compressed Video on ATM Networks", *Proc of IEEE INFOCOM'93*, pp919-926, 1993.

[29] L.A. Rowe, K.D. Patel and K.Liu, "MPEG Video in Software: Representation, Transmission, and Playback". in *Proceedings of IS&T/SPIE 1994 International Symposium on Elec. Imaging Sci. & Tech.*, 1994.

[30] Streamworks form Xing Tech. (www.xingtech.com)

[31] Tokuda, "Operating System Support for Continuous Media Applications", *Mutlimeda Systems*, Edited by J.F. Koegel Buford, Addison-Wesley Publishing Company.

[32] VDOLive Palyer form VDOnet (www.clubvdo.net)

[33] Vogel, Brigitte Kerherbve, Gregor von Bochmann and Jan Gecsei, "Distributed Multimedia and QoS: A Survey", *IEEE Multimedia*, volume 2, number 1, pp.10-18, 1995.

[34] Wai Hon-Kee, "Priority Feed back Mechanism with Quality of Service Control for MPEG Video System" *MSc Thesis*, Department of Computer Science, Hong Kong Baptist University, August 1999.