Multimed Tools Appl (2008) 36:303–324
DOI 10.1007/s11042-007-0147-2

Streaming of scalable h.264 videos over the Internet

Aylin Kantarcı

Published online: 27 June 2007

Abstract To investigate the benefits of scalable codecs in the case of rate adaptation problem, a streaming system for scalable H.264 videos has been implemented. The system considers congestion level in the network and buffer status at the client during adaptation process. The rate adaptation algorithm is content adaptive. It selects an appropriate sub-stream from the video file by taking into account the motion dynamics of video. The performance of the system has been tested under congestion-free and congestion scenarios. The performance results indicate that the system reacts to congestion properly and can be used for Internet video streaming where losses occur unpredictably.

Keywords Scalable coding · H.264/AVC · Content-aware streaming · Rate adaptation · QoS management

1 Introduction

Internet is a heterogeneous platform where available bandwidth may fluctuate dynamically in a wide range from a few kilobits per second to a few megabits per second and where hosts may have different display resolutions and system resources. Statistics indicate that streaming video content on the Internet, which was initially designed for data transmission and communication among computers, has increased tremendously in recent years [12]. Due to the timely delivery and high bandwidth requirements of streaming video objects, video compression and streaming technologies have great responsibility in providing smooth playback and the best available video quality to the users. In a typical streaming system, the video source performs *QoS management* by continuously monitoring the status of the network and by initiating an adaptation process to match the video rate to the available network bandwidth in case of congestion [8, 9]. In recent years, QoS management

A. Kantarcı (✉)
Computer Engineering Department, Ege University, Bornova 35100 Izmir, Turkey
e-mail: aylin.kantarci@ege.edu.tr

has received great interest from both industry and academics. In general, adaptation techniques follow one of the two major paradigms, namely *switching among multiple nonscalable bitstreams* and *streaming with a single scalable bitstream* [23].

In switching among multiple nonscalable bitstreams approach, a video sequence is encoded into several non-scalable bitstreams at different bit rates. Adaptation is performed dynamically by switching among these bitstreams. Streaming with a single scalable bitstream approach on the other hand, allows for partial decoding or transmission of a single bitstream consisting of multiple substreams. For example, when network bandwidth is limited, the server sends a set of substreams that matches the currently available network capacity. Similarly, to meet the requirements of clients with diverse display resolutions, computational and memory capabilities, the decoder may select appropriate substreams from a single stream that contains multiple versions of the video. Using scalable streams instead of switching among multiple bitstreams is more advantageous for several reasons. A few of these advantages can be listed in the following:

- In the case of multiple nonscalable bitstreams, when a change in the channel bandwidth is detected, switching is deferred until the arrival of a nonpredictive frame (I frame) of the selected bitstream to avoid drifting errors. This is because of the fact that nonscalable codecs assume that the coded video is completely received and properly decoded. However, the fundamental assumption in the scalable codecs is that decoded quality depends on the capabilities of the clients and error/loss status of the network. Hence, it is impossible to know the decoded quality at the encoding time [18, 22, 23]. Being developed under completely conflicting assumptions; signal processing facilities employed in the single scalable bitstream approach make it possible to switch to a substream with lower quality without having to wait for an I frame.
- In the case of nonscalable coding approach, different versions of a video are kept in separate files, which results in large storage space requirements. In the case of scalable coding approach, substreams of different bit rates are kept in only one file. This approach has less storage space requirement because; a lower quality bitrate constitutes a part of substreams with higher bitrates.
- In the case of nonscalable coding approach, a large metafile for each version of video is required to keep the offset of each random access point (I frames) to be able to switch the target bitstream at the correct point in the temporal domain. Therefore, an offline analysis is required to be performed to extract meta data for the positions of I frames when a new video is added to the system [8, 9]. On the other hand, scalable coding approach uses one bit stream and therefore, eliminates the requirement of storing offsets of I frames.
- Scalable coding approach makes application development and maintenance easier. Applications can be developed in shorter time with less effort and with shorter program code.

Scalability is attained by a layered coding approach which distributes the video information over a number of layers. The most fundamental information to reconstruct the video is placed on a layer which is called the *base layer*. Other layers, i.e. the *enhancement layers*, provide additive information to increase video quality. An enhancement layer requires a base layer and lower enhancement layers to be able to be decoded. The more layers the decoder receives, the better qualities it will achieve [22]. For video data, there are three dimensions of scalability: *Temporal scalability* means that frame rate of the video can be

adjusted. *Spatial scalability* refers to the capability of decoding video frames at different resolutions. *SNR (signal-to-noise ratio) scalability* allows for adjusting the target data rate in accordance with network status and system capabilities. Each scalability type results in different kinds of visual distortion depending on the content. In most cases, a single scalability type may not be suitable for the whole video. The best scalability option should be selected for each temporal segment depending on the content of that segment [20, 22, 29].

In this study, to investigate the benefits of scalable coding in the case of rate adaptation problem, a streaming system with adaptation facilities has been implemented and experiments under various network conditions have been conducted to demonstrate its performance. The system initiates adaptation process considering the network status and buffer status at the client side. The system is also content adaptive in the sense that it considers motion dynamics to select the best scalability type during adaptation decision. The videos in the system are encoded with the state of the art H.264 codec [14] that incorporates scalability extensions [15, 16]. Although official codecs such as MPEG-2 and MPEG-4 support scalability to some extent, scalable H.264 codec is distinguished from them in many respects. For example, it is not practical to use more than 2 or 3 layers with MPEG-2. The reason is that multiple compensation loops which lead to significant overhead and increase in encoder complexity are required due to the closed loop architecture of MPEG-2. Fine grain scalability (FGS) has been supported in MPEG-4. However, motion prediction with MPEG-4 FGS is based on the lowest quality base layer which has a negative affect on coding efficiency. For such reasons, scalability options of MPEG-2 and MPEG-4 are not widely deployed in video streaming applications [17, 23, 24]. Owing to its open loop architecture and incorporated state of the art signal processing facilities, H.264 codec with scalability extensions is superior to its official rivals by demonstrating the true notion of scalability as well as avoiding the drawbacks of the previous scalable codecs [7, 18]. It is stated to be the most distinguished official scalable codec in near future. Therefore, upon the considerations of the goals of this study, it has been realized that scalable H.264 codec has been the most suitable candidate among all scalable codecs.

It should be noted that no real streaming system that is based on scalable H.264 videos has been reported in the literature yet. Among the previous studies on scalable coding and streaming [2, 5, 21] are worth to consider. In [2], a rate distortion optimized framework to stream scalable bit streams of 3 D wavelet video has been proposed and its performance is measured via simulation. In that study, several parameters such as the desired playout deadline, transmission rate and network characteristics are taken into account to minimize the distortion in the reconstructed frames. The difficulty arises in determining some of these parameters in the actual Internet environment. For example, the proposed method uses target forward payload rate which depends on the available bandwidth which is difficult to measure in a connectionless network. The main concern in [5] is error resilience and error recovery. In this study, a scalable system with two layers in which the base layer is coded with H.263 is used in experiments. The performance of the system is measured via simulation as in [2]. The work in [21] is based on content adaptive video encoding with MPEG-4 fine granular scalability (FGS). In this work, the quality of visually important objects is increased with a novel FGS selective enhancement method. The proposed algorithm has been evaluated via simulation. Unlike the systems proposed in [2, 5, 21], the system developed in this study is unique in the sense that it has been carried out on actual network conditions. Secondly, it is the first reported streaming system based on scalable H.264 videos. Thirdly, the goal of this paper is different from those of the previous studies [2, 5, 21]. The primary concern of our work is to investigate and demonstrate how scalable videos can be used in the rate adaptation problem that is inherent in the Internet. Another contribution

of this study is that it combines the latest advancements in compression, processing and communication technologies to build an operational streaming system.

This paper is organized as follow: Section 2 is an overview on content adaptive rate control. The developed system is introduced in detail in Section 3. Performance results of the implementation are presented in Section 4. Finally, conclusions are drawn in Section 5.

2 Content adaptive rate control

In the case of rate adaptation against network congestion, the server should exploit scalability dimensions intelligently. Each scalability type introduces a different type of video distortion. Using only one scalability dimension throughout the whole video may increase visual distortion during some scenes. To maximize perceptual video quality, *video content* may be used as a decision criterion to switch among different scalability dimensions. For instance, if a scene has a lot of motion, i.e. it is dynamic, and data rate is to be reduced, SNR or spatial scaling can be the most appropriate method, because the details within a frame may not be so important in segments with high motion where the frame contents change rapidly. On the other hand, since there is little change between successive frames, temporal scaling can be more useful for scenes with low motion content. This makes details more visible and dropping frames does not degrade the perceptual quality very much [9, 10, 25].

In the family of MPEG standards, it is a practical way to analyze a video stream at the macroblock level to extract motion dynamics. In MPEG standards, three types of frames exist in an encoded video stream. *I* (intra) frames are obtained by exploiting the spatial redundancy within a frame (intra-coded) and independent of any other frames in the sequence. An I frame consists of only intra coded macroblocks. *P* (predicted) frames are coded with a previous I or P frame. These pictures are inter-coded and can be used as reference frames for B and P frames. A P frame may include both I and P macroblocks. *B* (bidirectionally predicted) frames are interpolated/predicted from previous and/or next reference frames. A B frame may contain I, P and B macroblocks [17, 28]. Among these macroblock types, I macroblock type indicates high motion content. When the number of intra macroblocks is high, there are rapid scene changes or new objects may have joined or an existing object may have moved out of the search area. On the other hand, a high number of B macroblocks interpolated with the macroblocks of previous and next frames indicates a great portion of the frame is similar to surrounding frames.

Therefore, counting the number of different types of macroblocks in each frame is a very helpful method in several types of applications such as video indexing and retrieval, shot detection, scene change detection and video editing. In [25], the percentage of interpolated macroblocks in B frames is used for measurement of motion in MPEG frames. A sequence of frames is categorized to have low motion content if it contains more than 45% interpolated B macroblocks. Otherwise, the sequence said to have high motion content.

Similar to previous standards, H.264 video frames consist of I, P and B slices. I slices contain only I (intra) macroblocks. P slices contain I and P (predicted) macroblocks. B slices consist of I, P and B macroblocks. P and B macroblocks are coded by prediction from one or more reference pictures. The difference between the two is that P macroblocks are predicted from the pictures from only one reference list, whereas B macroblocks are predicted from one or two references lists [6]. Similar to previous standards, a video segment with high number of I macroblocks implies high motion content. Unlike previous standards, macroblocks of P and B slices may be predicted by using more than one reference picture. P or B macroblocks derived from more than one reference pictures indicate

that the content of the current slice has a dependency on the surrounding frames which may imply low motion content.

3 The developed system

The architecture of the developed system is given in Fig. 1. The developed system has a client/server architecture. The server software accepts connection requests from clients, streams videos to clients and performs content adaptive QoS management considering network status and buffer fullness at the client site. Client software receives video packets, places them into a buffer and consumes video data from the buffer on a GOP basis. In relation to buffer management, it also tracks the buffer occupancy and notifies the server in case of an undesired change in buffer occupancy. In the following subsections, the components of the system will be introduced in detail.

3.1 Communication system and end system support

The protocol stack used in the system is given in Fig. 2. The developed system uses RTP for unicast data transfer. Network statistics are fed back to the server via an accompanying protocol called RTCP [19]. For the exchange of control messages, UDP is used. Scalable H.264 videos are packetized according to the guidelines given in the related Internet draft [27]. An important feature of H.264 is that it decouples the signal processing from transmission tasks by dividing the encoding process into two separate layers. The video coding layer (VCL) contains the signal processing functionality of the codec. The network abstarction layer (NAL) encoder encapsulates the slice output of the VCL encoder into network abstraction layer units (NALU). This approach allows for easy packetization of data without having to know about the structure of the video file and any change in the VCL layer results in minimal change in the packetization process. Maximum packet size

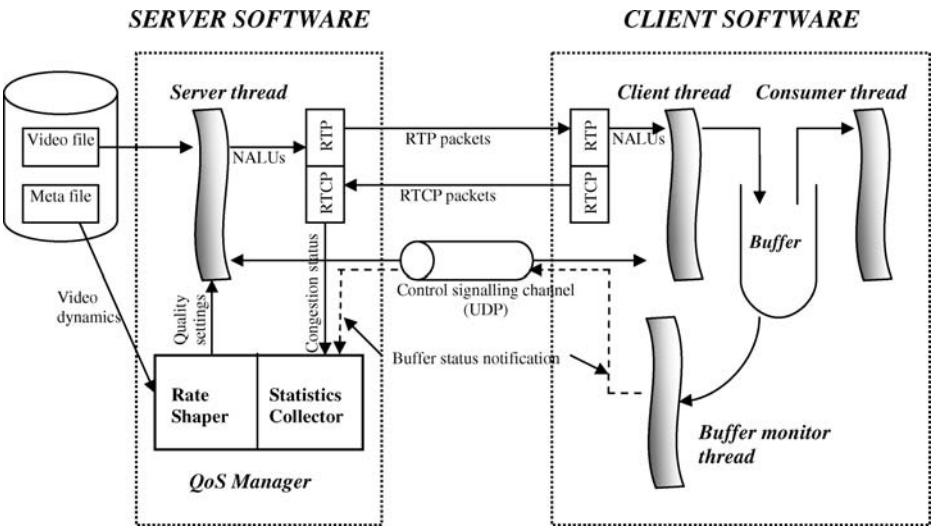

Fig. 1 System architecture

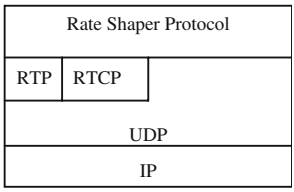

Fig. 2 Protocol stack for the developed system

is restricted by the LAN/WAN MTU. Among the packet types proposed in [27], Multi-time aggregation packets (MTAP) and fragmentation units (FU_B) are suitable for the developed system. Consecutive NALUs that can fit into a packet of maximum MTU size are encapsulated in a MTAP, whereas NALUs larger than the MTU size are fragmented into several FU_Bs. The rate control algorithm used by the *QoS Management Engine* regulates the transmission rate based on the loss rate information reported in RTCP reports sent by the receivers and tries to maintain the receiver buffer occupancy within a desired interval.

The performance of a streaming system depends on how well the resources of the end systems are managed in addition to the capacity of the transmission medium. For this purpose, facilities of POSIX standards have been extensively used in the system [4]. For example, to introduce real-time support, clock granularity has been decreased to nano-second level with the use of POSIX.4 real-time extensions. Additionally, various overheads of the operating environment have been tried to be minimized with this library. For example, to avoid the overhead of disk scheduling, video data are mapped to the memory once the video file is opened and locked into main memory throughout the streaming. Another POSIX standard employed to increase the performance of the system is POSIX 4.a standard which provides multithreaded programming support [11]. Multithreading allows for parallelism by activating threads that use independent resources at the same time. Multithreading paradigm has been extensively used in both server and client modules.

3.2 The server module

Main tasks of the server module are accepting connection requests of clients, streaming videos to clients and performing QoS Management. The server module has a multithreaded architecture. Various server tasks are assigned to distinct threads which are executed concurrently with the thread that serves the client's requests.

Clients transmit their connection requests over the control signal channel. If the requested video is available, a new thread is created to serve the request of the client and a RTP/RTCP channel is established between the client and the server. The server maps the video file to the memory and processes the user's request in two passes. In the first pass, bandwidth requirement of each substream is determined by performing a quick analysis on the video stream and stored in a table to calculate the transmission interval between consecutive packets during the second pass, in which the video is streamed to the client. Since compressed video induces VBR traffic on the network, constant packet interval value is not suitable for smooth delivery. In the second pass, upon switching to another bitstream, the table obtained in the first pass is used to calculate an approximation for the transmission interval between consecutive packets. During streaming, QoS Management Engine collects loss and buffer status statistics and initiates rate adaptation if necessary.

Let l be the index of the current substream, n_l be the expected number of packets required for the target substream for the whole video, Bytes[] is the array of bandwidth demands of substreams, packet_sizes[] is the array of approximate rtp packet sizes for video

substreams and t_l be the transmission interval between the consecutive packets in the target substream. Then, the following equations are used to calculate an approximation to t_l :

$$n_l = \text{Bytes}[l]/\text{packet\_size}[l]; \tag{1}$$

$$t_l = \text{video\_duration}/n_l \tag{2}$$

In the initial experiments, t_l had been calculated by using a single packet size equal to the maximum MTU size (=1,500 bytes). During the streaming of substreams with high quality in which FU_B packets are employed, approximations were more accurate because the actual packet sizes are close to maximum MTU size. On the other hand, actual packet sizes were smaller than the maximum MTU size for substreams with low quality substreams due to the restriction that a NALU which does not fit into a MTAP packet is not to be fragmented but to be placed in a new RTP packet as a whole. Therefore, it was more appropriate to use different packet sizes for each substreams to calculate t_l values more accurately.

3.3 The client module

The responsibility of the client module is to receive, to buffer and to consume video packets in real time. The pipelined nature of typical video players has been implemented by using multithreading paradigm. At the client site, one thread is responsible for receiving packets from the network and placing them into a buffer. Another thread consumes them from the buffer on GOP basis. Pipelining allows for concurrent execution of CPU and IO bound tasks, thereby increasing system throughput and avoiding packet loss at the network interface card. A third thread in the client software tracks buffer occupancy. This thread sleeps until an undesired condition related to buffer occupancy arises. Then, the sleeping thread wakes up and notifies the server about the important event related to buffer status. Thanks to the conditional variable and mutex provisions of POSIX 4 library, periodic polling of buffer status has been avoided.

3.4 QoS manager

One of the most important components of the server module is the *QoS manager*. This module is composed of two subsystems, namely the Statistics Collector and the Rate Shaper. The Statistics Collector subsystem collects loss statistics conveyed through RTCP reports and notification messages on the buffer status through the control signalling channel. The QoS manager evaluates those statistics and invokes the rate shaper module if rate adaptation is required.

The Rate Shaper module performs rate adaptation by switching to a proper substream considering motion dynamics. Two separate submodules, *Congestion_Responder* and *BufferNot_Responder* are executed in accordance with the reason of rate adaptation. The former module reacts to RTCP messages reporting the existence of congestion whereas the latter one reacts to buffer notification messages sent by the receiver.

In general, both types of messages may arrive simultaneously during congestion periods. In this case, congestion status takes precedence over buffer status because losses have more critical effects than delays. Otherwise, two consecutive rate reductions would be inevitable and quality would be decreased at a larger step than actually needed. In the following subsections; Congestion_Responder and BufferNot_Responder modules will be introduced in more detail.

3.4.1 Congestion_Responder

The QoS manager evaluates the loss statistics conveyed through RTCP reports to determine the congestion level at the network. Upon reception of an RTCP packet, the first task of the QoS manager is to determine whether the congestion is short or long termed. Short-term losses are transient and it may not be necessary to initiate rate adaptation during this kind of losses. On the other hand, long term losses are prolonging and they require rate adaptation. To distinguish between the two cases, loss rate is smoothed with a low pass filter slr=$(1-\alpha)$ slr+(α) llr, where slr is the smoothed loss rate, llr is the lost rate in the last RTCP report and α is a weight such that $0<\alpha<1$. Smoothed loss rates are used to determine the level of congestion in the system. The Congestion_Responder assumes the existence of the congestion classes given in Table 1.

These threshold values have been chosen such that the systems operates in a flexible manner without disruption and unnecessary quality degradations. Our experimental findings indicate that the thresholds given in Table 1 are suitable for the proper operation of the system. In the experiments with higher threshold values, the QoS manager acted late to initiate the proper actions and buffer drainings occurred, worsening the system performance. On the other hand, smaller threshold values made the system more conservative; initiating unnecessary rate adaptation actions and quality degradations.

The algorithm of the Congestion_Responder module is given in the pseudo code in Fig. 3. As seen from the Fig. 3, when the network is uncongested state, the sender probes for the available bandwidth by progressively increasing its sending rate in small steps. Quality enhancements are performed conservatively, because a sudden large increase in data rate may itself result in congestion. In the LIGHTLY_LOADED status, the prevailing quality is preserved. When the network is in the MEDIUM_CONGESTED state, video rate is decreased in small steps. In the HIGH_CONGESTION state, video rate is reduced in larger steps. Finally, in the SEVERE_CONGESTION state, video rate is decreased to the lowest possible value, by switching to the substream with the lowest quality.

3.4.2 BufferNot_Responder

Another event that invokes the QoS manager module is the arrival of control messages that convey information about the status of the buffer at the client site. The client thread places the received packets into a buffer for several reasons. Firstly, in an UDP based environment, packets may arrive out of order. Buffering enables reordering of packets at the application level. Secondly, the effects of jitter can be compensated by storing packets in a buffer for a given amount of time. Thirdly, buffering has an important role in preserving the continuity of the display. An important indicator of the performance of streaming applications is the number of gaps during display. To minimize the number of gaps, some amount of data is prebuffered prior to consumer thread starts execution and buffer occupancy is tried to be

Table 1 Congestion classes for the developed system

Congestion class	Smoothed loss rate interval (%)
NOT_CONGESTED	[0–2)
LIGHTLY_LOADED	[2–5)
MEDIUM_CONGESTION	[5–15)
HIGH_CONGESTION	[15–30)
SEVERE_CONGESTION	[30–...)

```
while (1)
{
 rtcp_p=wait_for_a_rtcp_packet();
 slr=Calculate_SmoothedLossRate(rtcp_p);
 con_stat=Determine_ConStat(slr);
 switch (con_stat)
 {
  case SEVERE: ADAPT (minrate);break;
  case HIGH: ADAPT(FAST_DECREASE);break;
  case MEDIUM: ADAPT(SLOW_DECREASE);break;
  case LIGHTLY_LOADED: break; //do nothing
  case UNCONGESTED: ADAPT(INCREASE); break;
 }
}
```

Fig. 3 Pseudo code for the Congestion_Responder module

kept within a reasonable interval while streaming proceeds. A prebuffering period of 15 s has been adequate for the developed system. Experiments show that the longer the pre-buffering period, the longer time it takes the buffer to drain but at the expense of viewers' dissatisfaction due to the waitings during the initial buffering period.

Buffer level remains constant as long as the input rate to the buffer equals to the consumption rate from the buffer. However, buffer occupancy drops for network or end system oriented reasons. In general, a sudden increase in delay is often followed by packet loss. Therefore, preventive measures may be taken by acting upon a notification of increasing delay. Monitoring buffer level is a practical way of commenting on network delay. Delays result in smaller input rates than consumption rates. Therefore, the decrease in buffer level may be the indicator of an upcoming congestion. In this case, it may be useful to decrease the video rate to reduce network load to help the prevention of the future congestion.

Buffer occupancy may also decrease even though the network is lightly loaded. This is due to the fact that the calculated transmission interval values between consecutive packets are approximations to the optimum values. The drift between the actual and optimum packet interval values manifests itself by a continuously and slowly decreasing buffer level. In this case, the server should decrease the packet interval to catch up with the optimum packet interval. Additionally, the client may decrease its consumption rate to help avoiding an upcoming buffer draining. On the other hand, buffer occupancy starts to rise when the network congestion is alleviated. In this case, the server increases video quality. Taking all these factors into consideration, the system uses the algorithm whose pseudo code is given in Fig. 4 upon the arrival of a buffer status notification message from the client. The BufferNot_Responder tries to keep the buffer level between two thresholds. The lower threshold, 10 s, is used to detect the danger of buffer underflow. The second threshold, 30 s, is used to warn the danger of an overflow, which is vital especially when the available memory is limited. The selection of the value of the high threshold depends on the available memory. The smaller the lower threshold, the higher the probability of buffer draining. The higher the lower the threshold, the more conservative the system, triggering rate adaptation longer before the probability of buffer draining occurs. If the buffer level falls below 10 s and the network is in the NOT_CONGESTED state, the first action of the server is to decrease the packet interval by 10%. Similarly, the client decreases consumption rate by 10% for 10 s. If buffer level continues to drop, video quality is decreased by rate adaptation. If the buffer level is above 30 s., the server stops sending video packets for 5 s so that the buffer level could fall below 30 s. When the buffer level is between the two thresholds, no

Fig. 4 Pseudo code for the BufferNot_Responder algorithm

```
scale=1;
while (1)
{
 buf_not=wait_for_a_buffer_notification_message();
 switch(buf_not.type)
 {
  case OVERFLOW: sleep(5); break;
  case UNDERFLOW:
          if (scale < 0.85) {
                //if decreasing packet_int does not stop
                //the decrease in buffer level; decrease quality
                ADAPT(DECREASE);
                scale=1;
          }
          else { // decrease packet interval by 10%.
                scale=scale*0.9;
                packet_int=scale*packet_int;
          }
  default: break;
 }
}
```

notification message is received from the client. In this case, provided that the network is in uncongested state, the Congestion_Responder module increases the video quality.

Both Congestion_Responder and BufferNot_Responder trigger the *Adapt* routine of the rate shaper which is responsible for rate adaptation (Fig. 5). Rate adaptation is performed by dropping or adding layers considering motion dynamics of video segments. When a new video is added to the system, it is analyzed at macroblock level on GOP basis to determine the dynamic and non-dynamic portions of the video. Starting and ending positions of each segment are stored in a file to be used in rate adaptation decisions during streaming. In the developed system, the algorithm given in Fig. 6 has been utilized to determine the motion level of the video segments. In dynamic scenes, spatial and SNR scaling result in less degradation in video quality whereas in static scenes, temporal scaling results in less reduction in visual quality. For the segments with medium level of motion, a balance between all scalability types is tried to be maintained. The threshold values in Fig. 6 have been chosen by visually inspecting a few videos and by considering similar studies in literature [10, 25]. It has been preferred to perform the motion analysis in off line manner due to the processing overhead of examining all macroblocks of GOPs. Otherwise, a separate thread would be needed to perform this analysis concurrently with the server thread. An online analysis would require that the nth GOP to be analysed at the latest during the transmission of the (n−1)th GOP. When the goal of designing streaming servers to serve as many clients as possible is considered, it is seen that the increased CPU load

Fig. 5 Pseudo code for the Adapt module

```
ADAPT (type)
{
if (Should_This_Request_Be_Processed())
{
 switch (type)
 {
  case FAST_DECREASE:  FAST_RATE_DECREASE();break;
  case SLOW_DECREASE:  SLOW_RATE_DECREASE();break;
  case INCREASE:  QUALITY_INCREASE(); break;
  case MINRATE: current_quality_level=0; current_temporal_level=0;
                Drop all FGS layers;
                break;
 }
}
}
```

Fig. 6 Pseudo code for the motion analysis step

```
MOTION_ANALYSIS_FOR_THE_CURRENT_GOP()
{
inta% = percentage of intra content;
predicted_P%= percentage of  P content predicted from more than one reference frame;
predicted_B%= percentage of  B content predicted from more than one reference frame;
if (intra% >= 20 %) scene_type = DYNAMIC;
else
        if  ((predicted_P% + predicted_B) < 45%) scene_type = STATIC;
        else scene_type = MEDIUM;
}
```

would be high, slowing down the server and resulting in a decrease in the number of concurrent requests to be served.

If there is severe packet loss, video rate is decreased sharply by initiating a FAST_RATE_DECREASE process (Fig. 7). To decrease the video rate sharply, all FGS layers are dropped at once. Additionally, quality layer and temporal level settings are decreased by one.

If the congestion is at medium level, video rate is decreased at smaller steps (Fig. 8). FGS layers are dropped firstly. If this does not help, quality is decreased by considering motion dynamics. During scenes with high and medium motion content, quality layers are dropped whereas during the scenes with low motion, temporal layer is reduced by one. Base temporal layer is 3 and 1 for scenes with medium and low motion content, respectively. In static video segments, when the temporal level equals to the base temporal layer at a given quality layer, current quality layer is dropped.

Figure 9 shows the algorithm executed when video quality is to be increased. A temporal level/temporal levels is/are added during scenes with medium or high motion content, respectively. Otherwise, a quality layer is added. The base temporal level for the dynamic scenes is one less than the maximum number of temporal levels at that quality layer. On the other hand, the base temporal level for the scenes with medium motion content is two less than the maximum number of temporal levels at that quality layer. After all quality and temporal layers are added, the quality is enhanced with FGS layers.

It should be noted that frequent quality adjustments may disturb the viewer. Additionally, some amount of time should elapse to observe the effects of an action of the adaptation module. Therefore, a conservative approach based on a hysteresis model has been followed to limit the interval between consecutive invocations of the adaptation module. This interval is 10 and 20 s in the case of quality degradations and quality increases, respectively.

4 Experimental work

The streaming system introduced in the previous section has been implemented with C language on Sun Ultra Sparc 5 workstations with Solaris 2.8 operating system. RTP library has been obtained from Lucent Technologies. POSIX.4 library and Pthread multithreading

Fig. 7 Pseudo code for the fast rate reduction algorithm

```
FAST_RATE_DECREASE()
{
        Drop all FGS layers
        Drop a quality layer
        Drop a temporal level
}
```

```
SLOW_ RATE_ DECREASE   ()
{
if (Drop_A_FGS_Layer ()  == 0)
 {
 switch (content)
 {
     case HIGH_MOTION:
                          current_quality_layer=current_quality_layer-1; break;
     case MEDIUM_MOTION:
                          current_quality_layer=current_quality_layer-1;
                          if (current_temporal_level  <BaseTemporalLevel_for_Medium_Motion)
                             current_temporal_level = BaseTemporalLevel_for_Medium_Motion;
                          break;
     case LOW_MOTION:
                          if (current_temporal_level > BaseTemporalLevel_for_Low_Motion)
                            current_temporal_level = current_temporal_level-1;
                    else current_quality_layer=current_quality_layer-1;
                          break;
 }
 }
}
```

Fig. 8 Pseudo code for the slow rate reduction algorithm. Drop_A_FGS_Layer() returns 0 when all FGS layers have been dropped

```
QUALITY_INCREASE()
{
 if (all quality and temporal layers have been added) Add_A_FGS_Layer();
 else
  {
    switch (content)
    {
     case HIGH_MOTION:
          if (current_temporal_level != highest_temporal_level of  the current quality layer)
                current_temporal_level = highest_temporal_level of  the current quality layer;
           else {current_quality_layer=current_quality_layer + 1;
                 current_temporal_level = max(
                                highest_temporal_level of  the current quality layer-1,
                                current_temporal_level);
                }
           break;
     case MEDIUM_MOTION:
           if (current_temporal_level != highest_temporal_level of  the current quality layer)
                 current_temporal_level = current_temporal_level+1;
           else{
                 current_quality_layer=current_quality_layer + 1;
                 current_temporal_level =max(
                        highest_temporal_level of  the current quality layer-2,
                        current_temporal_level);
                }
           break;
     case LOW_MOTION:
           if ((current_quality_layer == highest_ quality_ layer) &&
                 (current_temporal_level != highest_temporal_level of  the current quality layer))
                   current_temporal_level =current_temporal_level+1;
           else current_quality_layer = current_quality_layer +1;
           break;
    }
  }
}
```

Fig. 9 Pseudo code for the quality increase algorithm

library have been used to provide real-time support and concurrency, respectively. As shown in Fig. 10, the developed system has been tested on an Ethernet LAN network with a server workstation, a client Workstation and a PC running CLOUD™ software between them. CLOUD™ enables the PC to act as a router by forwarding packets sent from one workstation to the other and simulates actual WAN environment by dropping and delaying packets in accordance with a given loss and queuing delay pattern. The queuing delay values indicate the amount of time the CLOUD™ software keeps the video packets in its buffer. Note that there is another delay type called *transmission delay* which is obtained by dividing the amount of data to be transmitted by the bit rate of the channel. The transmission delay cannot be controlled in any way during the experiments http://www.shunra.com (web page for the CLOUD™ software).

YUV (4:2:0) videos have been coded with the JSVM software [15] to produce scalable H.264 videos. Two sample videos have been used to test the system. The scalability parameters and bandwidth requirement of these videos are listed in Table 2. As seen in Table 2, Video1 has less stringent bandwidth demand than Video2. According to [3, 15], two FGS layers have been found to be adequate for coding efficiency purposes. Therefore, two FGS layers have been associated with each layer. In the tables, "bit rate without FGS" columns refer to the range of bitrates without FGS layers, "bit rate with FGS" column refers to the range of bitrates when all FGS layers are included. When these columns are carefully examined, it can be seen that the network load induced by FGS layers is much higher than that of other layers.

Various experiments have been conducted to observe the behavior of the system. In one of the experiments, the system has been tested with a scenario in which the bandwidth is constrained to 1,544 Kbps. In another experiment, to simulate a congestion scenario, CLOUD™ is configured to drop and delay packets for approximately 60 s in accordance with a given congestion pattern. In another experiment, QoS Management module has been excluded from the system to observe the effects of the rate adaptation on the system performance.

In the following subsections, performance results for the test scenarios have been presented. In the experiments, the following parameters have been used to evaluate the performance of the system:

1. *Loss statistics*: Loss statistics presented in this section have been extracted from the fraction_lost field of RTCP receiver reports sent by the receiver. This field denotes the fraction of lost RTP packets since the last RTCP packet from the receiver. The system smoothes the loss rates with a loss pass filter to avoid unnecessary rate regulations due to momentary increase of loss rate values.
2. *Buffer statistics*: Buffer draining results in halts during display, displeasing the viewers. Therefore, buffer status at the client site is an important QoS metric in streaming

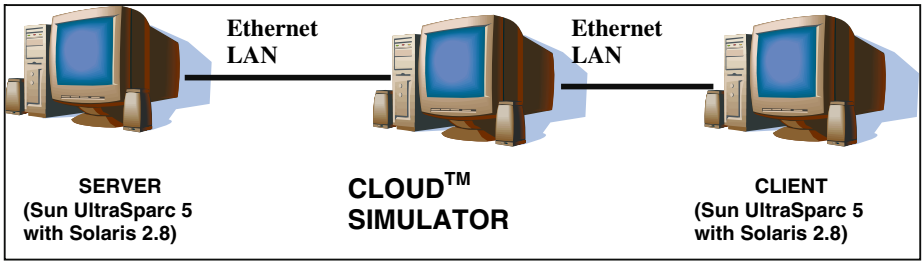

Fig. 10 The test bed used in the experiments

Table 2 Properties of videos used in the experiments

	Layer no.	Resolution	Frame rate (fps)	Bit rates without FGS (kbit/s)	Max bitrate (kbit/s)
VIDEO 1	0	176×144	3.75, 7.5	49, 57	182, 229
	1	176×144	0.9375, 1.875, 3.75, 7.5, 15	43, 47, 50, 59, 68	123, 162, 211, 276, 348
	2	352×288	0.9375, 1.875, 3.75, 7.5, 15	85, 98, 110, 129, 150	354, 467, 598, 756, 940
	3	352×288	0.9375, 1.875, 3.75, 7.5, 15	85, 100, 114, 138, 169	504, 706, 1,050, 1,474, 2,047
	4	352×288	0.9375, 1.875, 3.75, 7.5, 15, 30	86, 100, 115, 140, 173, 214	583, 859, 1,242, 1,768, 2,504, 3,137
VIDEO 2	0	176×144	7.5,15	217, 254	662, 952
	1	176×144	1.875, 3.750 , 7.5, 15	195, 210, 219, 257	416, 593, 878, 1,342
	2	352×288	1.875, 3.750, 7.5, 15, 30	462, 544, 621, 732, 832	1,216, 1,750, 2,514, 3,650, 4,583
	3	352×288	1.875, 3.750, 7.5, 15, 30	463, 546, 624, 736, 840	1,613, 2,487, 3,826, 5,972, 8,495
	4	704×576	1.875, 3.750, 7.5, 15, 30	597, 758, 926, 1134, 1353	2,652, 4,298, 6,784, 10,262, 15,629
	5	704×576	1.875, 3.750, 7.5,	59, 761, 931, 1,143,	3,477, 5,839, 9,527,

applications. The frequency of buffer draining depends on the initial buffering period, the bit rate of the stream and the bandwidth available between the sender and the client. Number of gaps per minute for various prebuffering periods has been given in Fig. 11. The smallest initial buffering periods with no gaps are 6 and 12 for Video1 and Video2, respectively. This is because Video2 demands more bandwidth than Video1. The higher the bandwidth requirement, the higher the transmission delay values and the higher the rate of decrease in buffer occupancy. Another video (Video3) with much less bandwidth requirement than Video1 and Video2 has also been used in the experiments on prebuffering duration. Experimental findings justify the relation between video bitrate and initial buffering duration.

Another importance of buffer statistics is that it is a practical way of commenting on the network delay. Measuring the network delay continuously may increase the load of the server.

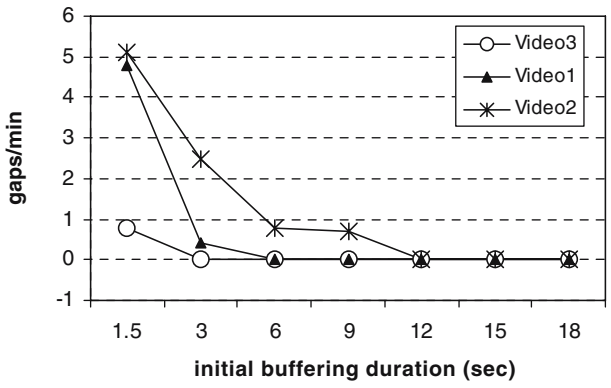

Fig. 11 Relationship between prebuffering time and gaps per minute

On the other hand, the conducted experiments reveal that there is a high probability that the when the delay increases, a drop in buffer level is observed. Therefore, buffer occupancy has been included in the performance parameters.

3. Quality settings: *Quality layer*, *temporal level* and *FGS layer* settings have been presented in the graphs that depict system performance. Quality layer reflects the spatial and SNR qualities, whereas temporal level and FGS layer are the indicators of temporal and SNR qualities, respectively.

Additionally, a practical parameter for perceptual quality has been introduced. There are objective video quality metrics reported in literature [1]. Among them, VQM is the most reliable one producing results with the highest correlation with the subjective quality metrics. However, its mathematical complexity is high and it requires the highest calculation capacity and effort among all other metrics. The simplest metric is the well-known parameter PSNR. However, it does not work well at low bit rates and when packet loss occurs. Secondly, it does not match well to the characteristics of the human visual system since it considers only the luminance component [1, 13, 26]. Thirdly, the same PSNR value during scenes with high and low motion dynamics does not refer to the same perceptual quality. Therefore, in this study, a practical parameter called *PerceptualQuality_Index* has been devised to demonstrate the perceptual video quality with respect to the best perceptual quality of a video segment. The PerceptualQuality_Index is a function of quality layer, Temporal quality, FGS layer settings and motion dynamics of a video segment.

$$\begin{aligned}\text{Current\_PerceptualQuality} = {} & \mathrm{w_q}*\ \text{Quality\_Layer} + \mathrm{w_t}*\ \text{Temporal\_Level} \\ & + \mathrm{w_{fgs}}*\ \text{FGS\_Layer}\end{aligned} \tag{3}$$

$$\begin{aligned}\text{Best\_PerceptualQuality} = {} & \mathrm{w_q}*\ \max\_\text{Quality\_Layer} + \mathrm{w_t}*\ \max\_\text{Temporal\_Level} \\ & + \mathrm{w_{fgs}}*\ \max\_\text{FGS\_Layer}\end{aligned} \tag{4}$$

$$\textit{PerceptualQuality\_Index} = \frac{\textit{Current\_PerceptualQuality}}{\textit{Best\_PerceptualQuality}} \tag{5}$$

In (3) and (4); w_q, w_t and w_{fgs} are the weight values which incorporate motion dynamics to the perceptual quality. Their values with respect to the scene type are given in Table 3. The weights have been selected such that the share of temporal scalability dimension has the highest share in dynamic scenes where as it has the least share in scenes with low motion. A

Table 3 Weight values for (3) and (4)

Motion type	w_q	w_t	w_{fgs}	Total
High	0.15	0.7	0.15	1.0
Low	0.35	0.3	0.35	1.0
Medium	0.25	0.5	0.25	1.0

balance between all scalability dimensions has tried to be established in scenes with medium level of motion.

4.1 System behavior under constrained bandwidth

In this experiment, CLOUD™ has been configured to limit the bandwidth between the server and client workstations to 1,544 Kbps. System behavior for Video2 has been presented in Fig. 12. In this experiment the initial quality has the highest settings (quality layer=5, temporal level=5 and FGS layer=2). Average bandwidth required at these settings is 31,402.1 kbps, which is much higher than the available bandwidth of 1,544 kbps. Therefore, as soon as the streaming starts, congestion at HIGH level occurs. The Rate Shaper responds by dropping all FGS layer, a quality layer and a temporal level. As seen in Table 2, the most suitable quality settings for the bandwidth of 1,544 kbps are (quality layer=4; temporal level=5 and FGS layer=0) and (quality layer=5; temporal level=4 and FGS layer=0). It has been observed that no loss occurred at these settings throughout the experiment. Since the network is uncongested, the QoS manager increases video quality to (quality layer=5, temporal level=5 and FGS layer=0). However, bandwidth requirement at this level is 2,465 kbps, which is higher than the available bandwidth. Therefore, momentarily loss occurs after a period of time this quality level is switched to. As shown in Fig. 12, the QoS manager adapts to the available bandwidth by decreasing video quality. Another interesting observation in this experiment is that, buffer level has increased up to the high threshold (=30 s) until t=175 s. The reasons for this increase are that the bandwidth demand of the prevailing quality settings is lower than the required bandwidth which does not lead to large transmission delays and that the estimated packet interval values for these settings are smaller than required. After consecutive notifications of the server about this situation, the server has ceased to send the video packets for 5 s. The effect of the pause has been observed as steadily decreasing buffer level during the rest of the experiment. Finally, as shown in Fig. 12, the perceptual quality has shown variation about 0.9 in accordance with the current quality settings and motion dynamics of video.

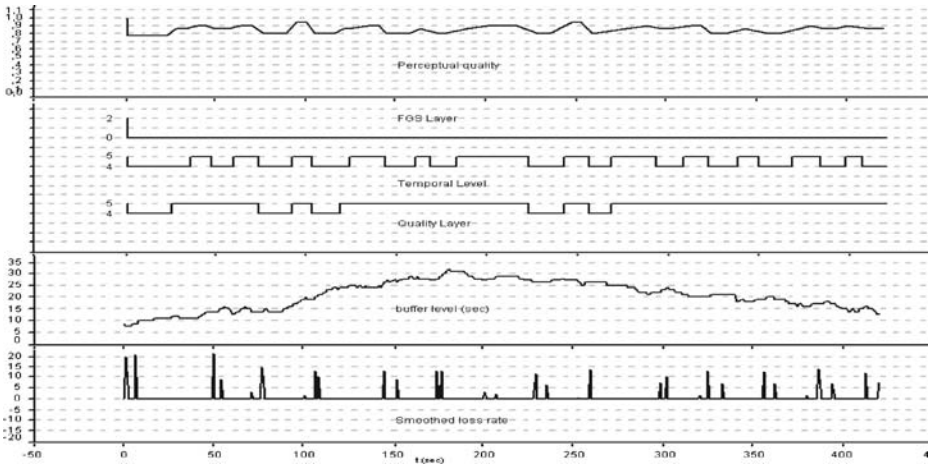

Fig. 12 System behavior when the bandwidth is constrained to 1,544 Mbps

4.2 System behavior under congestion

In this experiment, there is no constraint on the bandwidth. Instead, CLOUD™ has been configured to generate the loss, delay and jitter patterns given in Fig. 13. Performance results for Video1 and Video2 are presented in Figs. 14 and 15, respectively.

Figure 14 shows the behavior of the system under congestion when Video1 is transmitted. In this experiment, only buffer occupancy is considered in rate adaptation decisions until t=480 s when the congestion starts. In conjunction with the increases in video quality with the probing experiments which indicate sufficient network capacity, network load also increases. High network load results in high transmission delays. Buffer occupancy falls below the low threshold (=10 s) at t=90 s. In response, to trigger an increase in buffer level, the consumer thread at the client reduces its consumption rate by 10% for 10 s and the buffer monitor thread notifies the server about the status of client's buffer. In response, the server decreases packet interval by 10% since the network is not congested. After buffer occupancy rises above 10 s, video quality is enhanced with the addition of FGS layers. At t=130 s, video quality is at the highest level with the settings quality layer=4, temporal level=5 and FGS layer=2. Increased quality induces high network load and transmission delay, leading to the drop in buffer level below 10 s at t=140 s. It has been observed that at this quality level, the tricks with transmission and consumption rates have not been sufficient to increase buffer level above 10 s. FGS layers have been dropped one by one to alleviate the network load. After all FGS layers have been dropped, buffer level rises above 10 s at t=195 s. This pattern of buffer occupancy repeats itself until the congestion starts. CLOUD™ software has been configured such that the queueing delay values significantly increase prior to the packet loss which leads to quick drops in buffer level. It has been observed that the buffer occupancy falls below 5 s during congestion. Congestion status alternates between HIGH and MEDIUM states. Upon the arrival of the first RTCP report indicating the existence of congestion at HIGH level; all FGS layers, a quality layer and a temporal level are dropped. When the congestion is at MEDIUM level a quality or temporal level is dropped in accordance with the motion dynamics of the current scene. Lowest

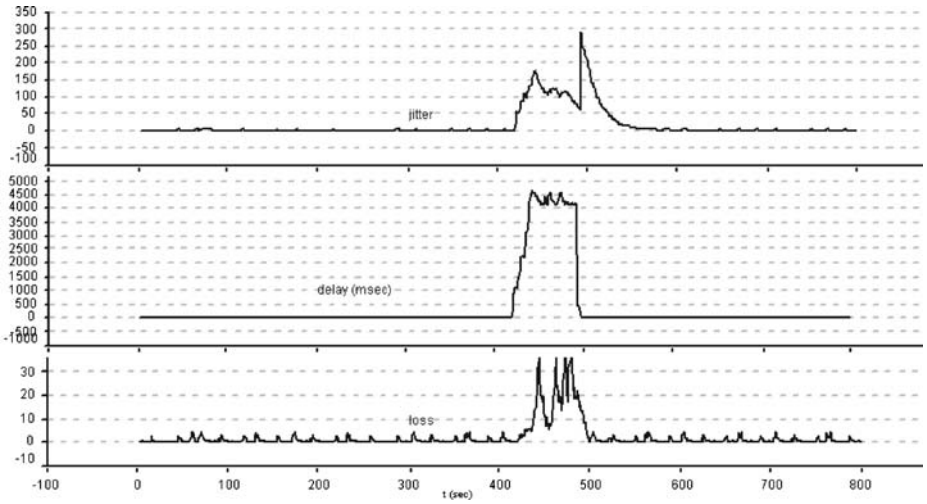

Fig. 13 Applied congestion in the experiments

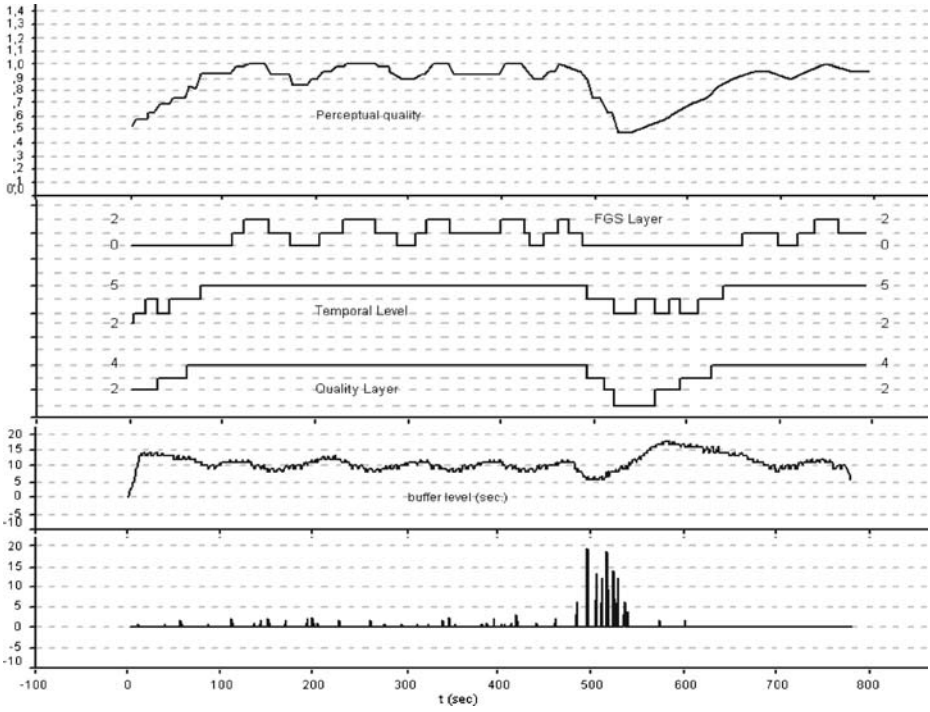

Fig. 14 System behavior for VIDEO1 when congestion occurs

quality during congestion has the settings with quality layer=1, temporal level=3 and FGS layer=0. At t=525 s, congestion ends. Low quality level imposes lighter load on the network and results in low transmission delays. Therefore, buffer level quickly increases up to 18 s. Meanwhile, video quality is enriched since the network is uncongested. In consequence, increase in buffer level stops. The system behavior is similar to the one observed prior to congestion. In this experiment, perceptual quality values vary in compliance with the prevailing quality settings throughout the experiment. It drops sharply during congestion and it is at the highest setting (=1) at the best quality settings quality layer=4, temporal level=5 and FGS layer=2.

Figure 15 shows the states of system variables during the experiment with Video2. As explained previously, Video2 imposes higher bandwidth demand than Video1. Although the same congestion pattern in the previous experiment has been applied, the graph in Fig. 14 depicts momentarily increase in congestion at t=110 s, t=145 s and a congestion period of MEDIUM level has been observed between t=180 s and t=220 s. When the graph is carefully examined, it is seen that packet losses occur in a short time period after FGS layers are started to be added. This shows that addition of a FGS layer increases the network load such that the available bandwidth becomes insufficient to meet the bandwidth requirement. As a consequence, the application itself causes congestion. During the congestion interval [180–220 sec], buffer level falls quickly below 10 s due to packet losses and high transmission delays. In this experiment, it has been observed that dropping only FGS layers has not been adequate to increase the buffer level above 10 s as in the experiment with Video1. Both quality layers and temporal layers have been dropped in

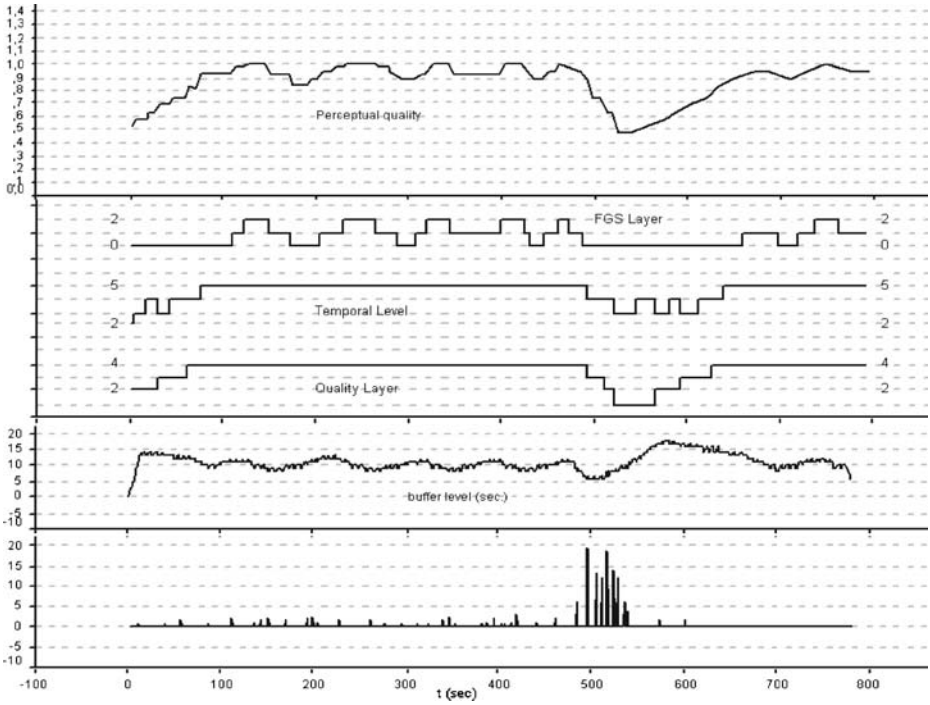

Fig. 15 System behavior for VIDEO2 when congestion occurs

order to increase the buffer level by reducing the transmission rate even if there is no congestion. At t=420 s, the configured congestion period starts. Due to the heavy congestion, video quality is quickly reduced down to the level with the settings quality layer=0, temporal level=1 and FGS layer=0. At t=480 s, both packet loss and queuing delay have been removed. Due to the prevailing low quality levels, network load and transmission delay is low. Consequently, buffer level rises above 10 s quickly and video quality is increased. As the network load increases, transmission delay starts to increase and buffer level starts to decrease. At t=520 s, it falls below 10 s. The tricks with consumption and transmission rates help to increase the buffer level above 10 s without reducing video quality. After quality layer and temporal level settings reach the highest values, quality is enhanced with the addition of FGS layers. However, in a short time after the first FGS layer has been added at t=550 s, the bandwidth demand increases above the network capacity and a momentarily increase in packet loss is observed. In sequence, adaptation module drops the currently added FGS layer to relieve the network status.

When the graphs in Figs. 14 and 15 are compared, it is seen that quality layer and temporal level settings have been managed to be kept at the highest values (quality layer=4, temporal level=5) throughout the experiment with Video1 as long as no congestion exists. Dropping only FGS layers has been adequate to keep buffer level around 10 s During the experiment with Video2, a video stream with much higher bandwidth demand than that of Video1, rate adaptation module had to drop quality layers and temporal layers in addition to FGS layers to keep buffer level above 10 s. Secondly, due to the excessive network load

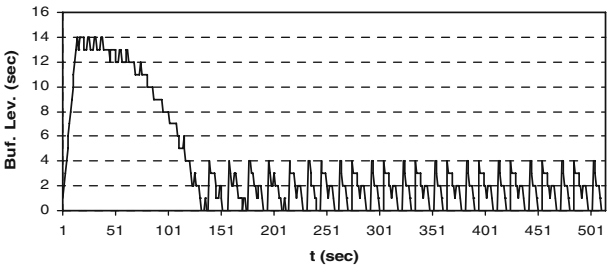

Fig. 16 Buffer status when the adaptation module has been omitted

introduced with FGS layers, we observed that the application itself caused congestion which was not configured with CLOUD™ software.

4.3 System behavior without the QoS management module

Finally, to make an evaluation of the advantages of the adaptation module to the system, we omitted it from the system and conducted experiments under congestion-free and congestion scenarios. Buffer occupancy graph for the experiment under the congestion free scenario is given in Fig. 16. As it is seen from the graph, video quality reaches the highest level at t=20 s. It remains constant for a short period and declines due to the large transmission delay resulting from the high network load induced by the best quality. It falls below 10 s at t=90 s. Since no action is taken to increase the buffer level, buffer level keeps falling and the first buffer drain occurs at t=130 s. In sequence, consumer thread stops consuming for 5 s to accumulate new packets in the buffer. After 5 s, 4 s of video have been stored in the buffer. The consumer thread wakes up and buffer drains quickly again. As a consequence, the consumer thread pauses again. This cycle is repeated throughout streaming. In another experiment with the congestion pattern given in Fig. 13, it has been observed that buffer drains more frequently during congestion periods.

When Figs. 14, 15 and 16 are examined, the value added to the system by the adaptation module will become clear. The adaptation module improves the satisfaction of the viewer by minimizing the number of gaps during display. Secondly, it reacts to congestion properly by reducing the load not to deteriorate network status.

5 Conclusion

In this study, a streaming system for scalable H.264 videos has been implemented. The system reacts to congestion in the network and drops in buffer level at the client side by switching to a lower quality substream. The system is content adaptive in the sense that it considers motion dynamics during rate adaptation process. The system has been tested with lossless and lossy network conditions. The experimental results show that the system properly reacts to changes in congestion level and buffer occupancy. The system is suitable to be used for Internet video streaming where losses occur any time unpredictably.

Currently, the system streams videos in unicast manner. This study investigates how scalable codecs can be used in rate adaptation problem. An important application class which scalable coding is very advantageous is multicast streaming. The next goal is to extend the system to be used for video multicast applications to investigate the benefits of scalable codecs in supporting clients with different resolutions and system resources.

References

1. Bistrom J (2005) Comparing video codec evaluation methods for handheld TV. Technical Report T-111.590. Helsinky University of Technology, Finland
2. Cahng C, Han S, Girod B (2004) Rate-distortion optimized streaming for 3-D wavelet video. Proceedings of IEEE International Conference on Image Processing Conference (ICIP 2004), Singapore
3. Description of core experiments in SVC, JVT-N025d0.doc (2005) http://ftp3.itu.int/av-arch/jvt-site/2005_01_HongKong, Hong Kong
4. Gallmeister B (1995) POSIX.4 Programmers guide programming for the real world. O'Reilley
5. Horn U, Stuglmüller K, Link M, Girod B (1999) Robust Internet video transmission based on scalable coding and unequal error protection. Image Commun 15(1–2):77–94
6. ISO/IEC 14496-10 (2004) H.264/AVC Standard documentation
7. Ji X, Xu J, Zhao D, Wu F (2004) Architectures of incorporating MPEG-4 AVC into three dimensional subband video coding. Proceedings of the Picture Coding Symposium (PCS-2004), USA
8. Kantarcı A, Tunalı T (2003) A video streaming system for MPEG-1 videos. J Multimedia Tools Appl Kluwer Acad 21:265–284
9. Kantarcı A, Özbek N, Tunalı T (2004) Rate adaptive video streaming under lossy network conditions. Signal Process Image Commun 19:479–497
10. Katou K, Zhao L, Sakauzowa S, Yamamato H (2003) An adaptive content-aware scaling for receiver driven layered video multicast. Proceedings of International Conference on Circuits, Systems, Computers and Communications, Korea
11. Lewis B, Berg DJ, SunMicrosystems Press (1998) Multithreaded programming with Pthreads. Prentice Hall, California, USA
12. Li M, Claypool M, Kinicki R, Nichols J (2003) Characteristics of streaming media stored on the Web. Technical Report WPI-CS-TR-03-18, CS Department, Worchester Polytechnic Institute (May)
13. Masry M, Hemai SS (2001) An analysis of subjective quality in low bit rate video. Proceedings of IEEE International Conference On Image Processing, Thessaloniki, Greece
14. Puri A, Chen X, Luthra A (2004) Video coding using the H264/MPEG-4 AVC compression standard. Signal Process Image Commun 19:793–849
15. Reichel HJ, Schwarz H, Wien M (2005) Joint scalable video model: JSVM0 http://ftp3.itu.ch/av-arch/jvt-site/2005_04_Busan/JVT-O202.zip, JVT-N020.doc, JVT Documentation, April
16. Reichel HJ, Schwarz H, Wien M (2005) Joint scalable video model: JSVM3, http://ftp3.itu.ch/av-arch/jvt-site/2005_07_Poznan/JVT-P202.zip, JVT Documentation, July
17. Richardson IG (2002) Video codec design, developing and video compression systems. Wiley, England
18. Santa_Cruz D, Maestroni D, Zilliani F, Reichel J, Tubara S (2004) Improved scalable MCTF video codec using a H.264/AVC base layer. Proceedings of the Picture Coding Symposium (PCS-2004), USA
19. Schulzrinne H, Cosner S, Fredric R, Jacobson V (1996) RFC 1889: RTP: a transport protocol for real-time applications. Network Working Group, IETF
20. Schwarz H, Marpe D, Wiegand T (2004) MCTF and scalability extension of H.264/AVC. Proceedings of the Picture Coding Symposium (PCS-2004), USA
21. van der Schaar M, Li Q, Boland L (2001) Internet video streaming with FGS. Proceedings of ISAS, La Havana, Cuba
22. Seok JM, Lee KH, Suh DY (2002) Effective streaming technology of a layered encoding video application supporting QoS mechanism in the Internet. Proceedings of the 2002 International Technical Conference On Circuits/Systems, Computers and Communications (ITC-CSCC-2002), Thailand
23. Sun X, Wu F, Li S, Gao W, Zhang Y (2004) Seamless switching of scalable video bitstreams for efficient streaming. IEEE Transactions on Multimedia, Special Issue on Video Streaming 6(2):291–303
24. Taubman D (2003) Successive refinement of video: fundamental issues, past efforts and new directions. International Symposium on Visual Communications and Image Processing (VCIP2003), SPIE 5150: 791–805
25. Triphathi A, Claypool M (2002) Adaptive content-aware scaling for improved video streaming. Proceedings of the 2nd International Workshop on Intelligent Multimedia Computing and Networking (IMMCN), USA
26. Wang Z, Banerjee S, Jamin S (2003) Studying streaming video quality: from an application point of view". Proceedings of IEEE Multimedia 2003, Berkeley
27. Wenger S, Hasnnuksela MM, Stockhammer T, Westerlund M, Singer D (2005) Request for comments: 3984, RTP payload format for H.264 video. February
28. Wiegand T, Sullivan GJ, Bjontegaard G, Luthra A (2003) Overview of the H.264/AVC video coding standard. IEEE Trans Circuits Syst Video Technol 13(7):560–576
29. Wu D (2001) Scalable video coding and transport over broadband wireless networks. Proceedings of IEEE, Special Issue on Multi-Dimensional Broadband Wireless Technologies and Applications 89(1):6–20

Aylin Kantarci received the B.Sc., M.Sc. and Ph.D. degrees from Ege University, Izmir, Turkey, in 1992, 1994 and 2000, respectively. She is an assistant professor at the Department of Computer Engineering at Ege University. Her current research issues include adaptive video streaming, video coding, operating systems, multimedia systems and distributed systems.