Architecture for efficiently streaming stored video using TCP

G.Srinivasa Rao, P.Satish Kumar, K.Devi Prasad, M.Supraja, G.Vinay Kumar, V.S.V.S.Murthy, M.Praveen Kumar, D.Rajesh, S.Lavanya, P.Ananda Deepthi, Ch.Pavan Satish

GITAM University

Abstract
TCP (Transmission Control Protocol) with its well-established congestion control mechanism is the prevailing transport layer protocol for non-real time data in current IP (Internet Protocol) networks. It would be desirable to transmit any type of multimedia data using TCP in order to take advantage of the extensive operational experience behind TCP in the Internet. However, some features of TCP including retransmissions and variations in throughput and delay, although not catastrophic for non-real time data, may result in inefficiencies for video streaming applications. In this paper, we propose an architecture which consists of an input buffer at the server side, coupled with the congestion control mechanism of TCP at the transport layer, for efficiently streaming stored video in the best-effort Internet. The proposed buffer management scheme selectively discards low priority frames from its head-end, which otherwise would jeopardize the successful playout of high priority frames. Moreover, the proposed discarding policy is adaptive to changes in the bandwidth available to the video stream.

Key words:
Video streaming, Congestion control, Adaptive frame discarding, explicit congestion notification, Differentiated services

1. Introduction

Transmission of high quality video over the IP (Internet Protocol) networks has become commonplace due to recent progresses in video compression and networking disciplines, the development of efficient video coders/decoders, the increasing interest in applications such as video on demand, videophone, and video conferencing, and the ubiquity of the Internet. However, that are certain technical challenges to be overcome for efficiently transmitting video over IP networks; see for example the references [1] and [2] for an introduction to the topic. These challenges stem from the mismatch between the strict bandwidth, delay, and loss requirements of the video applications and the best-effort current Internet, which was originally designed around data applications that can tolerate loss and delay. Moreover, the instantaneous bandwidth available to a certain user or application changes in all time scales because of the very dynamic nature of the Internet, making the problem evens more challenging. These characteristics of the Internet led to the rise of network-adaptive video applications for providing smooth playout at the receiving client.

This paper addresses the problem of TCP-friendly on-demand streaming of temporally scalable stored video over the Internet using server-side adaptive frame discarding. In a stored video-on-demand system, the server prestores the encoded video and transmits it on demand to a client for playout in real time. The client buffers the data and starts playout after a short delay in the order of seconds (called the playout delay and denoted by T_p). We assume a fixed $T_p$throughout the paper as opposed to the adaptive playout schemes where the client buffering delay is varied with respect to the network conditions [3],[4]. It is this tolerability to larger playout delays that distinguishes the stored video streaming problem from other video networking applications like video phony, video conferencing, and live video streaming. It is also very desirable that once the playout begins, it should be able to playout without any interruption (i.e., smooth playout) until the end of the video streaming session. Moreover, such a transmission strategy should not jeopardize the data flows on the same network path which use TCP as their transport protocol, which is referred to as the "TCP-friendliness" requirement [5],[6],[7].

For network-adaptive video transmission over IP networks, the server adapts its video injection rate into the network to the instantenous available bandwidth in the network. Several mechanisms are proposed for rate adaptation including stream switching as in the SureStream technology provided by RealSystem G2 [8],[9], rate-adaptive video encoding/transcoding [1], or joint use of scalable coding (i.e., layered coding) and rate

Manuscript received January 5, 2010
Manuscript revised January 20, 2010

shaping via server-side selective frame discard [10]. Bitstream switching does not offer a fine granularity since there are only a few bitstreams available among which the streaming server can switch. Rate-adaptive encoding is more appropriate for live video streaming or interactive video applications as opposed to the stored video streaming problem we discuss in this paper. In our work, we therefore focus on rate adaptation using scalable encoded bitstreams. Scalable video codecs generate two or more bit streams, one carrying the most vital video information, called the Base Layer (BL), and the others carrying the residual information to enhance the quality of the base layer, which is referred to as the Enhancement Layers (EL) [11]. If there is a single EL, then the corresponding scalable coding is called 2-layer. Several scalable video-coding techniques have been proposed over the past few years for real-time Internet applications in the form of several video compression standards such as MPEG-2/4 and H.263/H.264 [11],[12],[13],[14]. The types of scalability which are defined in these standards can be categorized as temporal, spatial, SNR, and object (only for MPEG4) scalability; In these structures, base and enhancement layers are precoded at encoding time, and therefore their rates cannot be adjusted at transmission time. Therefore, server-side selective frame discard mechanisms are proposed for rate adaptation of scalable video. These discard mechanisms intelligently decide to drop some EL frames with the goal of increasing the overall quality of the video by taking network constraints and client QoS requirements into consideration [10]. The more recent Fine Grain Scalability (FGS) coding in which the enhancement frame can be encoded independently with an arbitrary number of bits and the bit rate can thus be adjusted at transmission time for finer granularity is left outside the scope of the current paper. We limit the focus of this paper by using a 2-layer temporal scalability video encoding scheme provided by H.263 version 2 (H.263+) [13] although we note that our results also apply to other 2-layer scalable video encoding schemes.

Besides network adaptivity, another challenging issue for the stored video streaming problem over the Internet is to provide inter-protocol fairness. TCP (Transmission Control Protocol) is the de-facto transport protocol for data in the current Internet. TCP is designed to offer a fully reliable service which is suitable for applications like file transfers, e-mail, etc. On the other hand, the alternative transport protocol UDP (User Datagram Protocol) used by many current streaming applications does not possess congestion control. Consequently, when UDP and TCP flows share the same link, TCP flows reduce their rates in case of a packet drop. This leaves most of the available bandwidth to unresponsive UDP flows leading to starvation of TCP traffic in case of substantial UDP load. Some believe that the current trend in using UDP as the transport layer without congestion control can lead to a congestion collapse of the Internet due to the rapid growth of such applications like Internet telephony, streaming video, and on-line games [5]. Taking into consideration the dominance of TCP in today’s Internet traffic, it is therefore desirable that the throughput of a video streaming session be similar to that of a TCP flow under the same network circumstances (i.e., two sessions simulatenously using the same network path). Such a mechanism is called TCP-friendly and TCP friendly schemes need to be designed to be cooperative with TCP flows by appropriately reacting to congestion [5]. There are a number of TCP-friendly congestion control algorithms which have recently been proposed, such as the rate-based RAP (Rate Adaptation Protocol), equation-based TFRC (TCP-Friendly Rate Control) [6],[7], and window-based BCC (Binomial Congestion Control). The transmission rates of the proposed TCP-friendly algorithms are generally smoother than that of TCP under stationary conditions at the expense of reduced responsiveness to changes in the network state (e.g., a new session arrival/departure to/from the bottleneck link). Moreover, these TCP-friendly mechanisms do not provide reliable transfer as TCP does, making them more suitable for real-time applications. DCCP, the Datagram Congestion Control Protocol, is a new transport protocol being developed by the IETF that provides a congestion-controlled flow of unreliable datagrams. TCP-like congestion control without reliability and the equation based TFRC [7] form the basis for the two congestion control profiles ID 2 and ID 3, respectively, in the DCCP protocol suite The stored video streaming problem over resource constrained networks, like the Internet, has attacted the attention of many researchers. Given network bandwidth and client buffer constraints, a dynamic programming algorithm with reportedly significant computational complexity is developed for the optimal selective frame discard problem in [10] as well as several heuristic algorithms. However, this study is unable to accomodate the bandwidth variability patterns of the Internet since the network bandwidth is assumed to be fixed and a-priori known. On a similar ground, rate-distortion optimization-based video streaming algorithms have been developed, that obtain scheduling policies for both new and retransmitted frames using stochastic control principles but the proposed methods are relatively complex and their feasability remain to be seen. The reference [16] considers a practical frame dropping algorithm for MPEG streams over best-effort networks but they neither use a TCP-friendly congestion control algorithm nor they take into account the deadlines of frames. A dynamic frame dropping filter for MPEG streams is proposed in a network environment where the available bandwidth changes dynamically but this work also lacks the TCP-friendliness component. A number of studies focus on streaming video using new TCP-friendly transport protocols [7] while others employing TCP itself. One common objection to use of TCP for streaming applications is the fully reliable service model of TCP through retransmissions. While delays due to retransmissions may not be tolerable for interactive applications, the service model for TCP may not be problematic for video on demand applications. Moreover, the use of ECN (Explicit Congestion Notification) allows TCP to perform congestion avoidance without losses, limiting further the potential adverse effect of the TCP service model.

In this paper, we propose a stored video streaming system architecture which consists of an input buffer at the server side coupled with the congestion control scheme of TCP at the transport layer, for efficiently streaming stored video over the best effort Internet. The proposed method can be made to work with other transport protocols including DCCP but our choice of TCP in the current paper as the under-

lying transport protocol stems from the following reasons:

- Slowly-responding TCP-friendly algorithms perform reasonably well in terms of video goodput in stationary conditions. However, responsiveness is especially critical in the core of the Internet today which appears to be operating in the transient rather than in the stationary regime due to the large session arrival and/or departure rates to/from the network. On the other hand, TCP congestion control has a well-established responsiveness to changing network state and may be more appropriate in rapidly changing environments.
- TCP with its original congestion control but with its full reliability feature replaced with selective reliability would be a more appropriate fit as a transport protocol for the underlying problem but the standards in this direction have not finalized and are still evolving. We note that TCP's insistence on reliable delivery without timing considerations would adversely affect the performance of the system under packet losses especially for (near) real-time applications (e.g., applications requiring short playout delays). In this paper, we study the regimes for which TCP performance for stored video streaming is acceptable but also identify regimes for which TCP performs poorly and a new transport protocol would be needed.
- TCP is currently used for streaming applications in order to get through some firewalls that block UDP traffic.
- The choice of TCP as the transport protocol eliminates the unnecessary burden on the application-level designer by providing congestion control at the transport layer.
- Another key advantage related to providing congestion control at the transport layer (i.e., TCP) rather than "above UDP" is that the proposed scheme can make use of the services provided by the standard-based Explicit Congestion Notification (ECN) mechanism which provides a means of explicitly sending a "congestion experienced" signal towards the TCP sender in TCP acknowledgment packets. We note that explicit feedback significantly reduces the losses in the network and is therefore particularly useful in scenarios such as video streaming where the frequency of retransmissions due to losses is to be kept at a minimum.

In our proposed architecture, the buffer management scheme selectively discards low priority frames from its head-end which otherwise would jeopardize the successful playout of high priority frames. Moreover, the proposed discarding policy is adaptive to changes in the bandwidth available to the video stream. Contrary to many of the previously proposed adaptive transmission algorithms, the proposed Selective Frame Discard (SFD) strategy is simple and is easily implementable at the application layer by allowing additional information exchange between the transport layer and the application layer. Moreover, our proposed server-side frame discarding algorithm only needs to know the playout delay $T_p$and several network related variables which are made available by using the services of TCP and the playout buffer occupancy does not need to fed back to the server in this proposed scheme. Our simulation results emonstrate that scalable stored video can

efficiently be streamed over TCP with the proposed adaptive frame discarding strategy if the client playout delay is large enough to absorb the fluctuations in the TCP estimation of the available bandwidth. We also study the impact using Explicit Congestion Notification (ECN) in the network in terms of attained video quality. Finally, we compare the proposed edge-based server-side frame discarding solution with the core-based Differentiated Services (Diffserv) Assured Forwarding (AF) Per-Hop-Behavior (PHB) architecture in the context of stored video streaming and identify regimes in which the former architecture outperforms the latter.
The rest of the paper is organized as follows. In Section 2, the proposed architecture including the scalable coding model and the selective frame discard schemes are presented. The simulation platform and the numerical results are given in Section 3. We conclude in the final section.

2. Video Streaming Architecture

In this section, we first describe our video encoding model and then present the details of the proposed input buffer management scheme based on selective frame discarding.
The main goal of scalable coding of video is to flexibly support a heterogeneous set of receivers with different access bandwidths and display capabilities. Furthermore, scalable coding provides a layered video bit stream which is amenable to prioritized transmission. In this paper, we assume that the stored video is encoded into two layers, the BL and the EL, using the Reference Picture Selection mode of H.263 version 2 [13],[14]. In this structure, the BL is composed of Intra (I) and anchor P (predicted) frames whereas the EL is composed of the remaining P frames. P frames in the EL are estimated using the anchor P frames or I frames in the BL where anchor P frames are chosen using the Reference Picture Selection mode. Throughout the rest of this paper, we will denote the base layer frames by H (High- priority), and enhancement layer frames as L (Low-priority). A schematic diagram of the employed scalable video coding structure is shown in Figure 1. We leave the study of different temporal scalability models and other video coding standards for future research but we believe that the proposed architecture is applicable to other 2-layer scalable video codecs.

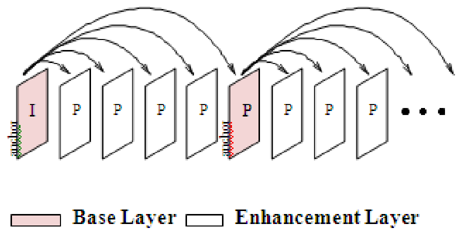

Fig. 1. Base and enhancement layers in temporal scalability mode

2.2 Selective Frame Discarding

As stated in the previous section, we assume that video encoders generate H- and L- frames. If the available network bandwidth cannot accommodate the transmission of all frames, then it would be desirable to discard to discard some of L- frames on the behalf of H-frames. While making a L-frame discarding decision ,our goal is to maximize the number of transported L-frames subject to the constraint that the loss rate for the H-frames would be minimal. In this definition, a loss refers to a missed frame at the client either because the frame is not transmitted by the server or is transmitted but partially/completely lost in the network or the frame is received by the client but after its deadline. For this purpose, we propose an input buffer implemented at the application layer of the sender which dynamically and intelligently discards L-frames from its headend and this scheme is depicted in Fig. 2.
We use the RTP/TCP/IP protocols stack in this study. We propose in this architecture that the stored video frames arrive at the input buffer at a frequency f = 1/T frames per second, which is the frame generation rate of the underlying video session. These frames wait in the input buffer until they reach the headend of the buffer and a decision is then made by the Selective Frame Discard (SFD) block whether the corresponding frame should be passed towards the transport layer or is simply discarded. In cases of discard, the SFD block will make subsequent discard decisions until an acceptance decision is made. When a frame is accepted by the SFD module, it is segmented into video packets (or RTP packets) of length at most L where we fix L to 1 Kbytes in this study. In our simulation studies, QCIF videos are encoded at around 30 dB quality and a typical video packet can carry 1-3 P-frames depending on the compression efficiency of the frame (i.e. high/low motion) and a

typical I-frame can be transported by 2-3 video packets. Video packets of accepted frames are first placed in the partial frame buffer which is then drained by the TCP layer. We suggest that whenever a TCP packet begins to take its first journey towards the network, the TCP layer immediately retrieves a packet from the partial frame buffer if the buffer is nonempty. Otherwise, it queries the SFD module to make an acceptance/rejection decision on the head-end frame. The acceptance/rejection decision is made as follows: The decision epoch for the ith frame is denoted by $t_i$irrespective of the outcome of the decision. The waiting time or the shaping delay in the input buffer for frame i, denoted by $D_{i,S}$, is the difference between $t_i$and the injection time for the i^{th}frame to the input buffer. Let $D_{i,N}$denote the network delay for the ith frame injected into the input buffer. Recalling that frames are generated by the encoder at integer multiples of T , the injection time for the ith frame to the input buffer will be t_0+ iT , where $t_0$is the injection time of the 0th frame. The ith frame will then wait in the input buffer for $D_{i,S}$seconds and the SFD module will make an admit/discard decision for the ith frame at time epoch $t_i=^4 t_0+ iT + D_{i,S}$. If the ith frame is admitted by the SFD module into the transport layer then that frame will be delayed an additional Di,T CP and Di,N seconds in the TCP buffer and in the network, respectively. It is clear that the ith frame must arrive at the receiver before its playout time $t_0+ D_0,N^+T_p+ iT$ where $T_p$is the initial buffering time of the playout buffer which starts accumulation as soon as the frame 0 arrives. So the following inequality should be satisfied for every accepted frame i > 0 for its succesful playout:

$$D_{i,S} \leq T_p - (D_{i,N} - D_0,N^{)} - Di,T\,CP \qquad (1)$$

In the above inequality, $D_{i,S}$and $T_p$are known to the SFD module, however one needs to find estimates for the last two terms on the right hand side of the inequality. In this study, we suggest to estimate the one-way network delay difference $\Delta_i = D_{i,N} - D_0,N$ using the TCP Timestamps option (TSopt) in TCP headers. In the TCP Timestamps Option, while transmitting packet m, the sender puts the transmission instant timestamp in the *TSval* (Timestamp Value) field. After receiving packet m, the receiver generates an acknowledgement packet denoted by ack m, by setting its *TSval* field with the current time of the receiver and by copying the *TSval* field of packet m to the *TSecr* (Timestamp Echo Reply) field of ack m. In this way, the SFD module will have an estimate of the one-way network delay difference using the TCP timestamp option for the last acknowledged TCP packet before time $t_i$when it needs to make a decision for frame i. On the other hand, the last term Di,T CP is not known in advance but is relatively small compared to $T_p$unless there are TCP losses because of the mechanism described for initiating a data transfer from the application layer into the TCP layer. We therefore introduce a safety parameter α, $0 < \alpha < 1$ to account for the errors due to inaccuracies due to estimations to be used in the inequality (1) as follows. In order for an admission decision for frame i to take place, the following new inequality should be checked by the SFD block:

$$Di,S \leq \alpha(Tp - \Delta i) \qquad (2)$$

The inequality (2) can be used to select which frames to discard for nonscalable video but it needs to be modified for layered video. This modification is studied next.

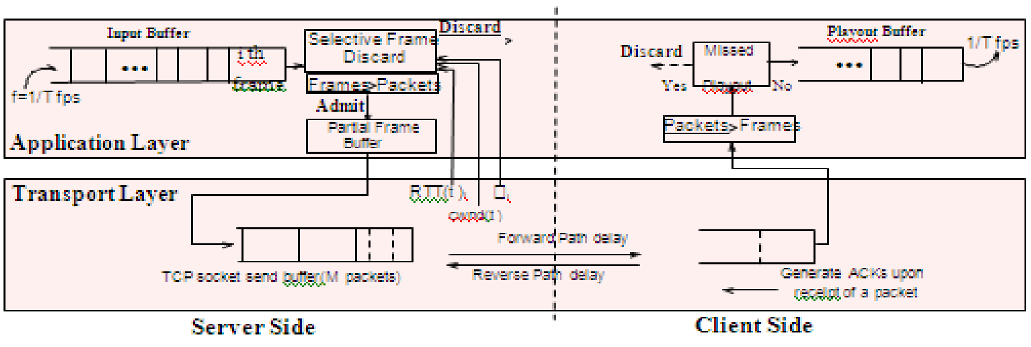

Fig. 2. Proposed stored video streaming architecture

2.3 Static and Adaptive Selective Frame Discard Algorithms

We propose to use two different safety parameters $\alpha_L$and $\alpha_H$for the L-frames and the H-frames, respectively, for preferential treatment for H-frames. Such a treatment is possible by choosing $\alpha L < \alpha H$. This choice makes αL not only a safety parameter but also a prioritization instrument. We summarize the general SFD algorithm at decision epoch $t_i$in Table 1.

The choice of the algorithm parameters $\alpha_L$and $\alpha_H$are key to the success of the proposed architecture. In Static SFD (SSFD), fixed $\alpha_L$and $\alpha_H$values are used throughout the video streaming session. However, such a fixed policy may not work well in all possible traffic scenarios. For example in cases where the instantenous available bandwidth is close to the the BL rate then the L-frames should aggressively be discarded (i.e., $\alpha L \rightarrow 0$) in order to minimize the loss probability of the BL frames. On the other hand, if the available bandwidth happens to be close to or exceeds the total rate of the BL and the EL frames, then the L-frames should conservatively be discarded (i.e. $\alpha L \rightarrow \alpha H$) . The very dynamic nature of the Internet may lead to significant variations in the available bandwidth even during the lifetime of a video session. Moreover the instaneous BL and EL rates for VBR encoded video may substantially deviate from their long-run average values. These observations lead us to an adaptive version of the SFD algorithm. For this purpose, we define C(t) as a smoothed estimate of the bandwidth available to the session at time t, where $Ci = C(ti)$ is simply the weighted average of $Ci-1$ and the instantaneous rate of TCP which is found by $cwnd_i/RT\,T_i$. Also we let $R_L(t)$ and $R_H(t)$ be the smoothed estimates of the EL and the BL, respectively, by monitoring the frame arrivals to the input buffer. We also let C, RL and RH denote the time averages of of the wave- Table 1

The pseudo-code for the SFD algorithm at time t_i

```
if ((frame i == L-frame) && ($D_{i,S}< \alpha_L(T_p- \Delta_i)$) ) {
    Admit();
} else if ((frame i == H-frame) && ($D_{i,S}< \alpha_H(T_p- \Delta_i)$) ) {
    Admit();
} else Discard();
```

forms C(t), $R_H(t)$, and $R_L(t)$, respectively. We then propose the simple Adaptive SFD (ASFD) scheme depicted in Fig. 3. We fix $\alpha_H$and use it only as a safety parameter ($\alpha_H$set to 0.7 in this study). The choice of $\alpha_L$is less straighforward: $\alpha_L$is zero when $C(t)< R_H(t)$, $\alpha_L$equals $\alpha_H$when $C(t) > R_H(t) + R_L(t)$ and it changes linearly within between these two end regimes. The notation SSFD(x) denotes the SSFD algorithm with α_H= 0.7 and $\alpha_L$set to x.

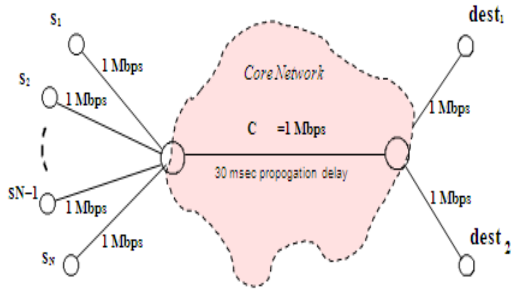

Fig. 4. The network topology used in the simulation studies

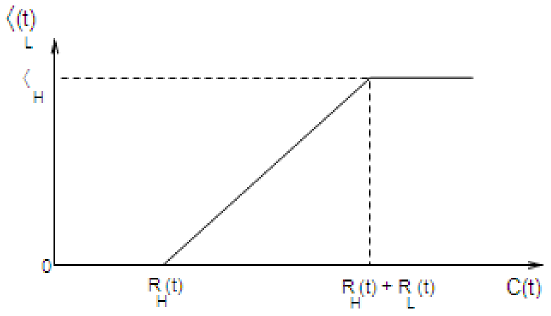

Fig. 3. Adaptive choice of $\alpha_L$in the ASFD algorithm

3. Simulation Results

In this section, we study the performance of the proposed stored video streaming architecture using simulation. We use ns-2 for simulations with a number of enhancements required for the video streaming architecture given in Fig. 2. We use the single bottleneck topology in Fig. 4 for all the simulation experiments. In all simulations, N video sessions (of length 780 seconds) share a single bottleneck link with capacity C_{tot}(set to 1 Mbps), where N will be varied to account for the variability of the available bandwidth to each user. The buffer management mechanism for the bottleneck link is assumed to be RED (Random Early Detect). We use the RED parameters (min_{th}, max_{th}, max_p) = (20, 60, 0.1) and the RED smoothing parameter set to 0.002 unless otherwise stated.

The first N/2 sessions are sinked at $dest_1$and the remaining ones at $dest_2$. Each video source employs TCP Reno with the same set of parameters and options and each source streams the same video clip. There is one tagged source we monitor among the N sources for PSNR (Peak Signal-Noise Ratio) plots. Each source starts streaming at random points in the video clip in order to prevent synchronization among the sources. Throughout the simulations, the bit rate of the VBR encoded video has substantial oscillations while the average rates are $R_L \approx$ 82.6 kbps and $R_H \approx$ 35.0 kbps (see Figure 5). Given that the original video frequency is f= 25 frames/sec, the two layer scalable video is composed of a single I and 9 anchor-P frames as the base layer for each two-seconds interval (i.e., Group of Pictures (GOP) duration). The remaining 40 are plain P frames that constitute the enhancement layer as given in Fig. 1. In our simulations, the average PSNR is used as the performance metric. For lost frames the concealment is done at the receiver by replicating the most recently decoded frame. Since we are using a temporally scalable bitstream, the PSNR of the received frames reflects the degradation in system performance due to losses only in the BL. Using PSNR for bothreceived and lost frames enables us to see the degradation in the system performance caused by both the L-frame and H-frame losses. In all of our experiments, the bottleneck link with capacity $C_{tot}$is shared among N sources where N $\in$ {6, .., 40} and the expected fair bandwidth share per flow, which is C $\approx C_{tot}/N$, changes in the range {25, . . . , 166} kbps.

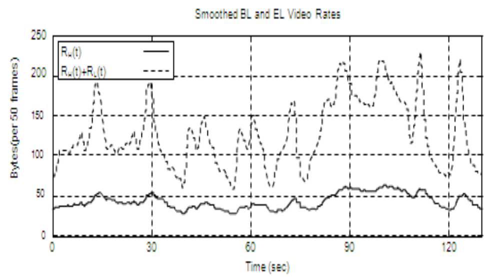

Fig. 5. Smoothed bit rates for the BL and EL for the layered video used in the simulations

In our first experiment, we compare and contrast the performance of the ASFD algorithm with the SSFD algorithm with three settings for $\alpha_L \in$ {0.05, 0.4, 0.7}. For this purpose, we vary the number of video sessions N and thus change the fair share of each session C $\approx C_{tot}/N$ and obtain the corresponding PSNR value for the SSFD and ASFD algorithms. The playout delay $T_p$is set to 5 sec in this study. The results are depicted in Fig. 6. The ideal curve is obtained by allowing the system to transmit and play all the scheduled frames, in other words for a given bandwidth it is assumed that there is enough playout buffering to tolerate the latency due to retransmissions and the video bitrate is properly matched to the constant available bandwidth in the network so that the scheduled frames never miss their playout times. In our simulations, the EL and/or BL frames are discarded sequentially for the computation of the ideal curve and the corresponding bitrate is calculated. The sequence used for discarding is the same for each GOP. The selection of a conservative SSFD policy (i.e., SSFD(0.05)) gives the best results for the heavy load case (i.e., C < 100 kbps) when compared to all other schemes. However, in the light load case when C gets close to or beyond R_L+ R_H, the PSNR performance of SSFD(0.05) degrades substantially compared to the less conservative policies SSFD(0.4) and SSFD(0.7). On the other hand, the adaptive version ASFD is robust with respect to the changes in the available bandwidth per user and it compares reasonably well with the best performing static policy in each case. The advantage of the ASFD is that the video server can find a policy very close to the optimal frame discarding policy using local measurements even when the available bandwidth per user changes significantly during the lifetime of the video session. This behavior can definitely not be obtained with static policies.

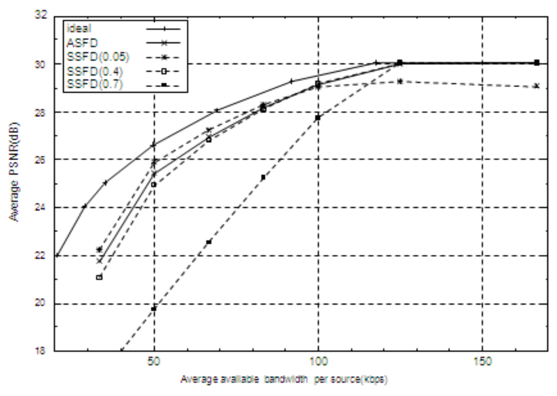

Fig. 6. Comparison of SSFD vs ASFD for the case T_p= 5 sec

In our second simulation experiment, we study the impact of the RED parameters on the ASFD performance.

The results are given in Fig. 7. The cases with three different RED configurations outperformed the drop-tail policy with the buffer size set to 120 packets. This observation can be explained by the fact that drop-tail buffer management causes synchronized losses and the resulting overshoots and undershoots in the resulting buffer occupancy yield substantial performance degradation relative to that of RED. We generally obtained quite robust results with RED but we also observed performance degradation with RED(10,30,0.1) in the heavy load case compared to the other two RED systems. This degradation is due to the relatively conservative choice of $min_{th}$and max_{th}in this system when a fairly large number of sources are multiplexed.

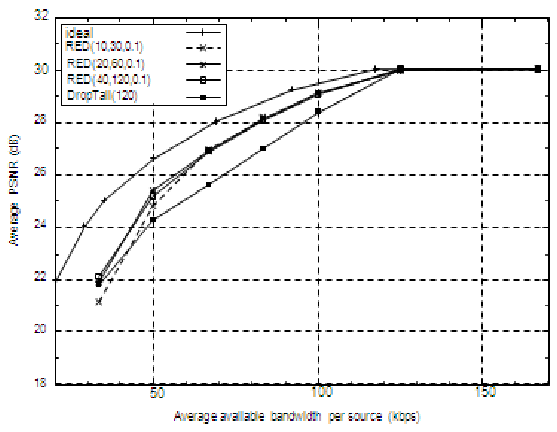

Fig. 7. Effect of RED parameters on ASFD performance with T_p= 5 sec.

In the third simulation experiment, we study the impact of using ECN for which the RED module at the bottleneck link marks the packets with the corresponding probabilities as opposed to discarding them. This congestion information is then fed back in the TCP acknowledgements via which the TCP sources adjust their window sizes. Since all TCP senders are using ECN and all respond to congestion before actually loosing a packet they tend to experience less the undesired data or timer driven loss recovery phases of TCP. This behaviour, as one might expect, leads to a singnificant performance improvement especially in congested network scenarios and for small initial playout delays. This situation is depicted in Fig. 8 in which Tp is set to 2 sec. and the performance of using TCP Reno without ECN and TCP Reno with ECN are shown in terms of the average PSNR values for varying C. For the heavy load case, the performance gain with ECN is remarkable (up to 2 db). The Tp = 5 sec. case is depicted in Fig. 9 for which the ECN gains are smaller compared to the T_p= 2 sec. case. For small playout delays, it is more likely that a larger percentage of the TCP's retransmissions arrive at the receiver later than their corresponding deadlines. With ECN, losses in the network are reduced and so are retransmissions. This is why the performance gain of ECN is more significant in cases with small playout delays. As shown in Fig. 8, T_p= 2 sec. of buffering cannot tolerate the timer driven retransmissions occuring in TCP, therefore a significant PSNR degredation is observed if ECN is not employed as compared to the T_p= 5 sec. case.

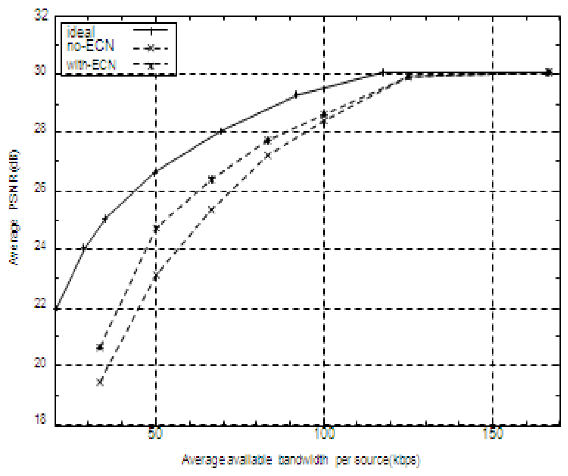

Fig. 8. Impact of ECN on streaming performance for ASFD with T_p= 2 sec.

In the fourth experiment, we study the impact of the playout delay $T_p$which is used in order to compensate for the oscillations in the video bit rate and available network bandwidth per user. The playout delay $T_p$is varied from 1 sec. to 30 sec. The PSNR curves saturate at around Tp = 15 sec. beyond which buffering only slightly improves the PSNR performance. For small T_p(i.e., T_p= 1 or 2 sec.), the playout delay is comparable to the delays encountered in TCP's data/timer driven retransmissions and a larger percentage of the network losses result in missed playouts and thus reduced PSNRs. With TCP, increasing $T_p$from 2 to 5 sec. increases the streaming performance substantially by up to 3 dB.

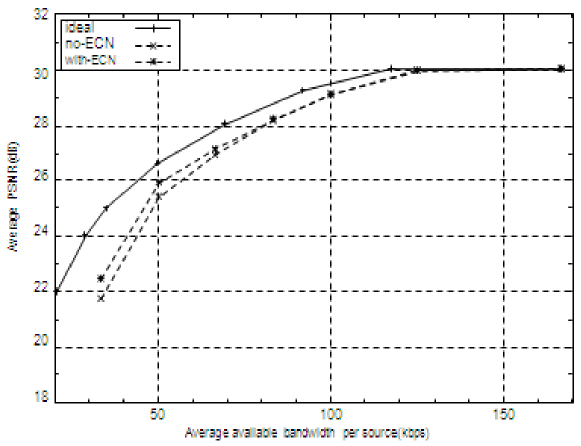

Fig. 9. Effect of ECN on streaming performance for ASFD with T_p= 5 sec

In our final simulation experiment, we compare the proposed edge-based server-side frame discarding solution with the core-based Differentiated Services (Diffserv) Assured Forwarding (AF) Per-Hop-Behavior (PHB) architecture in the context of stored video streaming and identify regimes in which the former architecture outperforms the latter. For the Diffserv scenario, we mark packets belonging to H- frames as AF11 and those of L-frames as AF12. We use Weighted RED (WRED) with the RED parameters (20, 60, 0.1) and (10, 30, 0.25) for AF11 and AF12, respectively. We do not impose the use of any traffic conditioner in this experiment but we make use of only the differentiated forwarding paradigm of Diffserv. We use UDP (User Datagram Protocol) for the transport layer for this scenario. We will refer to the combined scheme as Diffserv+UDP. which demonstrates that when the client playout delay $T_p$is small and comparable to one Round Trip Time (RTT), the Diffserv+UDP solution outperforms the proposed ASFD+TCP approach. However, when $T_p$is increased to 5 sec., then the ASFD+TCP solution gives better results than that of the Diffserv+UDP solution (see Figure 10). The reason for this behaviour is that when the client playout delay is large enough then the TCP sender can retransmit not ACKed packets without them missing their deadlines (as opposed to the T_p= 1 sec. case). Moreover, it is the application layer that intelligently decides on which frames to discard in ASFD+TCP by taking into consideration their playout deadlines. We're led to believe that when the playout delays are sufficiently large (i.e., T_p> 5 sec.) then the proposed edge-based adaptive approach is superior to the network-based Diffserv+UDP scheme which is static in its parameter settings and which is not aware of the playout deadlines.

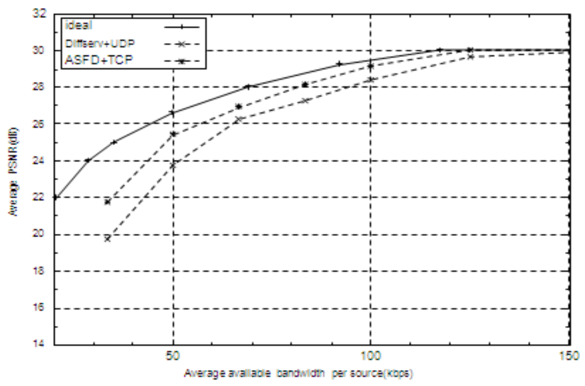

Fig. 10. PSNR plots using Diffserv+UDP and ASFD+TCP scheme for T_p= 5sec. scenario

4. Conclusions

Motivated by the extensive operation experience behind TCP, we propose in this paper an easily implementable stored video streaming system using TCP transport. The proposed system consists of an input buffer implemented at the application layer of the server coupled with the congestion control scheme of TCP at the transport layer. The proposed frame discarding strategy dynamically and intelligently discards low priority frames from its head-end. Moreover, it is adaptive to changes in the bandwidth available to the video stream. Our simulation results demonstrate that scalable stored video can efficiently be streamed over TCP with the proposed adaptive frame discarding strategy if the client playout delay is large enough to absorb the fluctuations in the TCP estimation of the available bandwidth. As expected, the use of Explicit Congestion Notification (ECN) in the network is shown to slightly improve the throughput especially in congested network scenarios and for small initial playout delays. Finally, we compare the proposed edge-based server-side frame discarding solution with the core-based Differentiated Services (Diffserv) AF PHB architecture and identify regimes in which the former architecture outperforms the latter. We show through a number of simulations that if the playout delay is sufficiently long (i.e., Tp > 5 sec.) then the proposed edge-based solution outperforms the core-based Diffserv solution whereas this relationship is reversed otherwise.

References

[1] D. Wu, Y. T. Hou, Y. Q. Zhang, Transporting real-time video over the Internet: Challenges and approaches, Proceedings of the IEEE 88 (12) (2000) 1855–1875.

[2] M. Civanlar, A. Luthra, S. Wenger, W. Zhu, Introduction to the special issue on streaming video, IEEE Trans. Circuits Syst. Video Technol. 12 (2001) 265–268.

[3] N. Laoutaris, I. Stavrakakis, Intrastream synchronization for continuous media streams: A survey of playout schedulers, IEEE Network Magazine 16 (3) (2002) 30–40.

[4] M. Kalman, E. Steinbach, B. Girod, Rate-distortion optimized video streaming with adaptive playout, in: Proceedings of ICIP, Vol. 3, Rochester, NY, 2002, pp. 189–192.

[5] S. Floyd, K. Fall, Promoting the use of end-to-end congestion control in the Internet, IEEE/ACM Transactions on Networking 7 (4) (1999) 458–472.

[6] J.Padhye, V. Firoiu, D. Towsley, J. Kurose, Modeling TCP Reno performance: A simple model and its empirical validation, IEEE/ACM Transactions on Networks 8 (2) (2000) 133–145.

[7] S. Floyd, M. Handley, J. Padhye, J. Widmer, Equation-based congestion control for unicast applications, in: ACM SIGCOMM, Stockholm, Sweden, 2000, pp. 43–56.

[8] A. Lippman, Video coding for multiple target audiences, in: SPIE Conference on Visual Communications and Image Processing, Vol. 3653, 1999, pp. 780–782.

[9] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, Y. A. Reznik, Video coding for streaming media delivery on the Internet, IEEE Trans. Circuits Syst. Video Technol. 11 (3) (2001) 269–281.

[10] Z.-L. Zhang, S. Nelakuditi, R. Aggarwal, R. P. Tsang, Efficient selective frame discard algorithms for stored video delivery across resource constrained networks, in: INFOCOM, Vol. 2, 1999, pp. 472–479.

[11] B. G. Haskell, A. Puri, A. N. Netravali, Digital Video: An Introduction to MPEG-2, Kluwer Academic Publishers, 1996.

[12] A. Puri, T. Chen, Multimedia Systems, Standards, and Networks, Marcel Dekker Inc., New York/Basel, 2000.

[13] Video coding for low bit rate communication, ITU-T Recommendation H.263 (February 1998).

[14] G. Cote, B. Erol, M. Gallant, F. Kossentini, H.263+: video coding at low bit rates, IEEE Trans. Circuits Syst. Video Technol. 8 (7) (1998) 849–866.

[15] A. Luthra, G. J. Sullivan, T. Wiegand, Introduction to the special issue on the H.264/AVC video coding standard, IEEE Transactions on Circuits and Systems for Video Technology 13 (7) (2003) 557–726.

[16] M. Hemy, U. Hengartner, P. Steenkiste, MPEG systems in best-effort networks, in: Packet Video Workshop, New York, 1999.

1. **Mr.G.Srinivasa Rao**, M.Tech,(Ph.D)., Sr.Asst.Professor. He has Submitted Ph.D thesis in M.U., Over 10 Years of teaching experience with GITAM University, handled courses for B.Tech, M.Tech. Research areas include Computer Networks And Data Communications. published 6 papers in various National and International Conferences and Journals.

2. **P.Satish Kumar** M.Sc., pursuing M.Tech(IT) from GITAM University.

3. **K.Devi Prasad** M.Sc., pursuing M.Tech(IT) from GITAM University

4. **M.Supraja** M.Sc., pursuing M.Tech(IT) from GITAM University..

5. **G.Vinay Kumar** M.Sc., pursuing M.Tech(IT) from GITAM University.

6. **V.S.V.S. Murthy** pursuing M.Tech(IT) from GITAM University.

7. **M.Praveen Kumar** pursuing M.Tech(IT) from GITAM University.

8. **D.Rajesh** M.Sc., pursuing M.Tech(CST) from GITAM University.

9. **S.Lavanya** pursuing M.Tech(IT) from GITAM University.

10. **P.Ananda Deepthi** pursuing M.Tech(IT) from GITAM University.

11. **Ch. Pavan Satish** pursuing M.Tech(IT) from GITAM University.