Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 00-060

Video Streaming across Best-Effort Wide-Area Networks with Controlled Quality Assurance Using Proxy Servers

Zhi-li Zhang and Yingfei Dong

December 05, 2000

Video Streaming across Best-Effort Wide-Area Networks with Controlled Quality Assurance Using Proxy Servers

Zhi-Li Zhang and Yingfei Dong

Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
{zhzhang,dong}@cs.umn.edu
Technical Report 00-60

Abstract

Video streaming across wide-area backbone networks using proxy servers is an important component of emerging global multimedia content delivery networks. In this paper we propose and develop a novel proxy-assisted video streaming technique—referred to as the staggered two-flow video streaming technique—for delivering videos across a best-effort wide-area backbone network from a central server to a video proxy server. The objective of the proposed video streaming technique is to provide controlled video quality assurance by taking advantage of the disk space at the proxy server, while circumventing the disk I/O bandwidth limitation at the proxy server in the mean time. Our technique is designed for stored videos that use compression schemes with inter-frame dependency, such as MPEG. Using the proposed staggered two-flow video streaming technique, a video stream is delivered in two separate substreams (referred to as flows): a flow containing the essential portion of the video (e.g., **I** frames) and the other containing the less essential portion of the video (e.g., **P**/**B** frames.) Both flows are segmented, and transmitted across the best-effort wide-area backbone network using different mechanisms. By delivering the **I** frame flow one segment ahead of time and caching the segment at the proxy server, we use a modified TCP with application-level throughput control (called *controlled TCP or cTCP*) to ensure the reliable transmission of the **I** frame segments while meeting their delivery deadlines. In contrast, the less essential flow containing the **P**/**B** frames is streamed unreliably (using RTP/UDP) in real time, and merged with the cached **I** frames by the proxy server using its memory buffer. We also develop rate adaptation and control schemes to deal with transient and persistent network congestion. Simulations are conducted to demonstrate the efficacy of the proposed staggered two-flow video streaming technique.

1 Introduction

Video streaming across wide-area backbone networks such as the Internet is an important component of emerging global multimedia content distribution networks. Proxy-assisted video delivery systems have been proposed and developed in both the research community [2, 5, 12, 14, 19, 16, 22, 24, 25] and industry [1, 8, 20]. Proxy servers that are strategically placed at the boundaries between wide-area backbone networks (e.g., the Internet) and local access networks are employed to assist in the delivery of stored videos from central video servers to a large population of geographically dispersed end users in a scalable manner. Proxy-assisted video streaming systems

offer several important advantages. For example, proxy servers can exploit their processing and buffering capabilities to provide network-wide video streaming and media control along the distribution tree in a coordinated but distributed manner. They can also utilize their potentially large disk storage space to prefetch video data, thereby significantly reducing the network resource requirements in the backbone wide-area network. Furthermore, because of their strategic positions inside an internetwork, proxy servers can take into account both the constraints of the underlying network environments as well as application-specific information in optimizing video transmission. Likewise, proxy servers can also leverage information about client end system constraints and QoS requirements to deliver video of diverse quality to clients.

Despite these important advantages, we also face some unique challenges in the design of such a proxy video streaming system. Although a proxy server provides additional disk space for caching or storing videos, its disk I/O bandwidth has been shown to be a major constraint that limits the number of videos which it can support [23, 24]. As a result, we cannot indiscriminately stage (i.e., pre-store) or cache entire videos at proxy servers. In other words, many videos (either in their entirety or in part) have to be streamed across the backbone wide-area network from a central video server. Because of the timing constraints, it is important to ensure continuous playback of videos from the central video server via proxy servers to end users so as to provide consistent (i.e., smooth) video quality. This problem is especially challenging when videos are streamed across a *best-effort* wide-area backbone network. Hence a key design question in building a proxy video streaming system is how to take advantage of proxy servers, while circumventing the I/O bandwidth bottleneck, to provide good video quality to end users.

To address these challenges, in this paper we propose and develop a novel video streaming technique — referred to as the *staggered two-flow video streaming* technique, for delivering videos from a central video server to a video proxy server across a best-effort wide-area backbone network. The proposed technique is developed for stored videos that are compressed using encoding algorithms with inter-frame dependence, such as MPEG. We divide each video stream into two separate substreams (referred to as flows): one flow contains the essential portion of the video (e.g., **I** frames), and the other contains the less essential portion of the video (e.g., **P**/**B** frames.) Both flows are partitioned into large *segments*, and transmitted across the best-effort wide-area backbone network using different mechanisms. The first segment of the flow containing the **I** frames is "*staged*" [16, 24] (i.e., pre-stored) at the proxy server, and the rest of the **I** frame segments are delivered *one segment ahead of* the corresponding **P**/**B** frame segments. In other words, the segments of the two flows are delivered in a staggered fashion. To ensure the reliable transmission of the **I** frame segments while meeting their delivery deadlines, we design a novel transport mechanism—the *controlled TCP (cTCP)* scheme which is a slightly modified version of TCP with application-level throughput control. The less essential flow containing the **P**/**B** frames is streamed unreliably in real time, using the standard RTP/UDP protocols. The **P**/**B** frames are merged with the cached **I** frames by the proxy server using its memory buffer.

With the help of the controlled TCP scheme, we address several interesting and challenging issues that arise in the development of our proposed staggered two-flow video streaming technique. One of them is the problem of bandwidth sharing between the reliable transmission of the **I** frame segments and the unreliable, real-time

delivering of the **P**/**B** frame segments. This problem affects the performance of our video streaming technique, because of the fact that the reliable **I** frame flow and the unreliable **P**/**B** frame flow of a video both follow exactly the same path from the central server to the proxy server across the wide-area backbone network. Using the controlled TCP scheme, we are able to control the interaction between the two flows, in particular, to minimize the impact of the reliable flow transmission on the packet losses experienced by the unreliable, real-time RTP/UDP flow. As a result, our video streaming technique yields more *stable* and *predictable* performance which is critical in providing consistent and controlled video quality assurance to end users. In addition, as will be briefly discussed in this paper, this performance predictability also enables us to provide a form of "application-level admission control" at the central server/proxy server, a function that can be very useful in an overlay multimedia content distribution network. To deal with both transient and persistent network congestion, we also develop rate adaptation and control schemes that take into account the application-specific information such as bandwidth requirements and frame types. Such rate adaptation and control schemes are important and necessary in delivering videos across best-effort networks. The efficacy of the proposed staggered two-flow video streaming technique and the associated control mechanisms are evaluated through extensive simulations. We demonstrate that, by intelligently taking advantage of the disk space at a proxy server, the staggered two-flow video streaming technique can indeed provide controlled video quality assurance, while circumventing the disk I/O bandwidth limitation at the proxy server.

The remainder of the paper is structured as follows. In Section 2 we present the problem setting, describe the proposed staggered two-flow video streaming technique, and illustrate how it can reduce the proxy disk I/O bandwidth requirement significantly. The controlled TCP scheme is developed in Section 3, and its performance is evaluated through simulations. In Section 4 we present rate adaptation and control schemes to deal with both transient and persistent network congestion at two levels: rate adaptation of the unreliable UDP flows within segments, and rate control at segment-level through dynamic segment size adjustment and dynamic switching between segments encoded at different quantization levels. The paper is concluded in Section 5.

2 Staggered Two-Flow Video Streaming Technique

In this section we describe the staggered two-flow video streaming technique, and illustrate how the proposed technique can help circumvent the I/O bandwidth constrains of proxy servers. The key issues in the development of this technique is briefly discussed at the end of this section. Section 3 and Section 4 are devoted to addressing these issues.

Figure 1 depicts a proxy-assisted video delivery system over a (rather simplistic) internetwork. At the core of this proxy video distribution system resides a (or several) *central video server(s)* with a large video repository. A collection of *(video) proxy servers* are strategically placed across the internetwork, typically attached to the gateway routers connecting the wide-area backbone network and the local access networks. These proxy servers, which are simpler in their functionality, assist the central server(s) in the distribution of stored videos to a large number of end users geographically dispersed at various local access networks. These proxy servers, together with

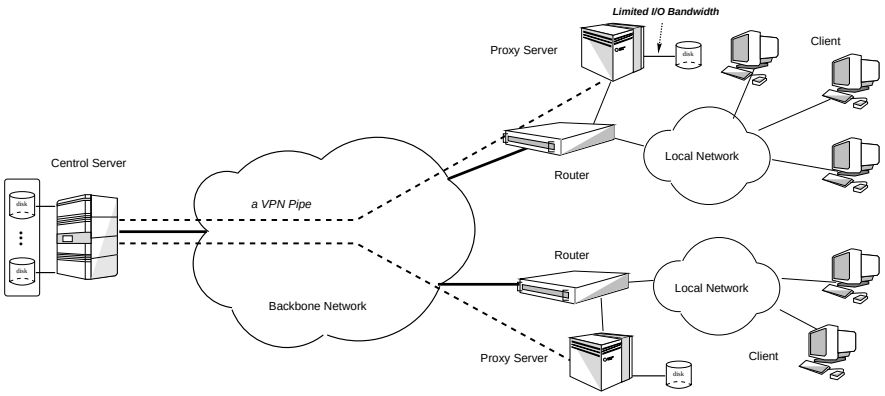


Figure 1: The system architecture.

the central server(s), form a virtual video distribution network over the internetwork.

The *staggered two-flow video streaming* technique we developed focuses on delivering videos from a central video server to a proxy server across a *best-effort* wide-area backbone network. By best-effort, we mean that the wide-area backbone network service provider does *not* provide any packet delay or loss guarantee for the delivery of videos across the network. However, we do assume that the videos from the central video server to a proxy server are transmitted across the wide-area backbone network through a "VPN (Virtual Private Network) pipe." (See Figure 1.) The (aggregate) bandwidth for the VPN pipe *may or may not* be provisioned with either a hard or a soft guarantee[1], depending on the service-level agreement between the network service provider and the content provider who owns the video proxy delivering system. The proposed technique is developed for stored videos that are compressed using encoding algorithms with inter-frame dependence. In this paper we use MPEG encoded video traces to illustrate how our proposed technique works.

The main objective of the proposed video delivering technique is to provide controlled video quality assurance by intelligently taking advantage of the disk space at a proxy server, while circumventing the disk I/O bandwidth limitation at the proxy server in the mean time. The basic ideas behind the staggered two-flow streaming technique are illustrated schematically in Figure 2. Each MPEG-encoded video is divided into two sub-streams (referred to as flows): one flow contains the essential **I** frames that are intra-frame coded; and the other flow contains the less essential **P**/**B** frames which depend on the **I** frames (and possibly other **P** frames)[2]. As we will explain shortly, the **I** frame flow will be transmitted *reliably* (using a modified version of TCP) across the wide-area backbone network, hence it is called the *reliable flow*; the **P**/**B** frame flow will be transmitted *unreliably* in real-time (using RTP/UDP) across the wide-area backbone network, hence it is called the *unreliable flow*. Both flows are partitioned into relatively large *segments* (with approximately equal length, measured in time, in the multitude of minutes). To take advantage of the proxy disk storage space, the first segment of the reliable **I** frame

[1] Such aggregate bandwidth guarantee can be provisioned, for example, in today's networks in a variety of manners such as leased line, ATM/Frame Relay virtual circuit, MPLS, or (statically configured) weighted fair queueing mechanisms.

[2] Depending on the system resource configuration (e.g., proxy disk space and I/O bandwidth), we can also divide a video stream in other manners For example, the reliable flow may contain not only the **I** frames, but also some or all of the **P** frames, whereas the rest of the frames (all **B** frames and maybe some **P** frames) is transmitted as the unreliable flow.

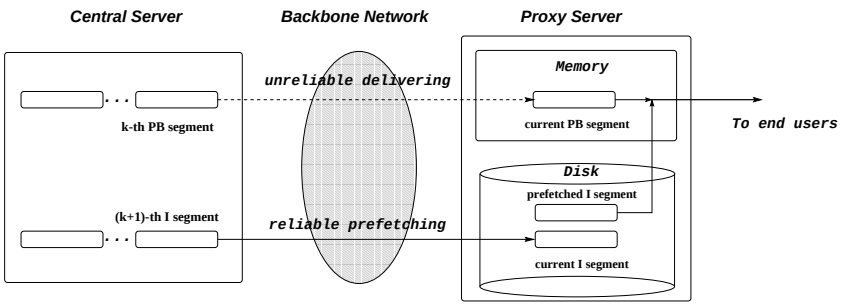


Figure 2: Staggered two-flow video streaming.

flow is staged (i.e., pre-stored) at the proxy server[3]. When an end user requests a video, the first segment of the unreliable (**P**/**B** frame) flow is delivered unreliably in real time from the central server to the proxy server. As the **P**/**B** frames in this segment are received, the proxy server merges them with the appropriate **I** frames that are retrieved from the local disk where the first segment of the reliable flow is pre-stored. The merged video stream is then streamed from the proxy server to the end user. At the same time the first segment of the unreliable flow is being transmitted from the central server to the proxy server, the second segment of the reliable flow is also being delivered from the central server to the proxy server. The second segment of the reliable flow is *cached* at the local disk, as it is not needed immediately. After the first segments of both flows are streamed to the end user, the second segment of the unreliable flow is transmitted unreliably in real-time from the central server to the proxy server, merged with the second segment of the reliable flow that is now cached at the proxy server. Again at the same time, the third segment of the reliable flow is delivered reliably from the central server to the proxy server and cached at the local disk of the proxy server. This process continues until the entire video is delivered to the end user.

From the above description, we see that the segments of the reliable and unreliable flows of a video are delivered in a *staggered* manner: for $n = 1, 2, \ldots$, the n_{th} segment of the unreliable flow is transmitted at the same time as the $(n+1)_{th}$ segment of the reliable flow is delivered from the central server to the proxy server. Note that since the reliable flow is delivered one segment ahead of time that it is needed, this provides us with sufficient time to recover any lost packets in the reliable flow during the transmission through TCP retransmission. In this way we ensure that all **I** frames are delivered reliably (i.e., lossless) from the central server to the proxy server across the best-effort wide-area backbone network. *This is one of the key features of our proposed staggered two-flow video streaming technique that enables us to provide some minimal video quality assurance to end users.*

By pre-storing, prefetching and caching only **I** frames of a video at the local disk of a proxy server, we also circumvent or alleviate the I/O bandwidth constraint of the proxy server. To see why this is the case, we note that there is only one **I** frame in each GOP (group of picture) group. For example, consider a video that is encoded at a frame rate of 24 frames/s with a GOP pattern **IBBPBBPBBPBB**. There is only one **I** frame every half a second. Since we only cache **I** frames in the local disk of a proxy server, on the average one **I** frame is retrieved

[3] To reduce the start-up latency, we can also pre-store a small "prefix" of the unreliable **P**/**B** frame flow at a proxy server [16].

Video trace	jurassic	MTV	silence	soccer	starwars	terminator	beauty	cnn	pbride	wizard
Avg Rate of Entire Video (Mb/s)	0.314	0.591	0.175	0.651	0.356	0.262	0.389	0.406	0.421	1.226
Avg Rate of **I** frames (Mb/s)	0.110	0.140	0.076	0.158	0.121	0.075	0.085	0.077	0.073	0.173
$\frac{Avg\ Rate\ of\ I frames}{Avg\ Rate\ of\ Entire Video}$	35%	24%	43%	24%	34%	29%	22%	19%	17%	14%

Table 1: Average rates in MPEG video traces.

video trace	jurassic	MTV	silence	soccer	starwars	terminator	beauty	cnn	pbride	wizard
An entire video	63	33	114	30	56	76	50	49	47	16
I frames only	181	142	263	126	165	265	235	259	273	115
Percentage Increase	187%	330%	131%	320%	195%	249%	370%	429%	481%	619%

Table 2: The number of video sessions can be supported on an Elite 9 disk.

every half a second from the local disk (from the already cached n_{th} segment of the reliable flow), while another **I** frame (from the $(n+1)_{th}$ segment of the reliable flow currently being transmitted to the proxy server) is written to the disk. This example intuitively illustrates that by caching only **I** frames in the local disk of a proxy server, we can potentially reduce the I/O bandwidth requirement significantly. To demonstrate the effectiveness of our staggered two-flow video streaming technique in reducing the I/O bandwidth requirement on a proxy server, we show some disk performance results using real video traces[4]. Table 1 lists the average bandwidth requirement of several MPEG-1 encoded videos as well as their respective average bandwidth requirement *when only I frames are considered*. In the third row of the table we see that in general the average bandwidth requirement of the **I** frames of a video is around 30% (or less) of the average bandwidth requirement of the entire video. Consider a Seagate Elite 9 disk with a SCSI-2 Fast interface. Based on the study in [23], this disk has an effective disk bandwidth of about 5 MB/s with a block size of 200KB. If we use this disk to cache all frames of a video, *Beauty and Beast*, which has an average bandwidth requirement of 0.389 Mb/s, only 50 video sessions can be supported simultaneously. Whereas if we only cache the **I** frames of the video which has an average bandwidth requirement of 0.085 Mb/s, the same disk can support 235 video sessions simultaneously, a 370% increase in performance. Similarly results are shown in Table 2, where we see that there is a significant increase for all video traces (from around 130% up to 619%). These results demonstrate that *for a given disk I/O capacity, the proposed staggered two-flow video streaming technique significantly increases the number of video sessions that can be simultaneously supported by the local proxy disk.*

So far we have illustrated the two major advantages of the proposed staggered two-flow video streaming technique for delivering videos across a best-effort wide-area backbone network: 1) By dividing a video into a reliable flow and an unreliable flow and staggering the transmission of these two flows, we ensure that all the essential video information (i.e., **I** frames) is transmitted reliably across the best-effort wide-area network in time, thereby providing some degree of minimal video quality assurance to end users. 2) By caching only **I** frames at the local disk of a proxy server, we circumvent or alleviate the proxy I/O bandwidth capacity constraints. However,

[4]The frame rate of the first six video traces *jurassic, MTV, silence, soccer, starwars* and *terminator* is 24 frames/s. The frame rate of video trace *beauty, cnn, pbride* and *wizard* is 30 frames/s.

the proposed staggered two-flow video streaming technique also poses some new interesting issues. Note that since both the reliable flow and the unreliable flow of a video follow the same path from a central server to a proxy server across the best-effort wide-area backbone network, they potentially compete for the bandwidth along the path. It is therefore important to control the interaction between the two flows, in particular, to minimize the packet losses experienced by the unreliable flow. This problem becomes especially acute when multiple videos are streamed through the same VPN pipe from the central server to the proxy server. Furthermore, in the case that the (aggregate) bandwidth of the VPN pipe is not guaranteed by the network service provider, the bandwidth available for streaming videos through the VPN pipe may change over time. To avoid drastic changes in the user perceived video quality, the rate at which video flows are delivered across the best-effort wide-area network must also be adjusted dynamically in a controlled manner, taking into account the application-specific information such as frame types. Due to the bursty nature of videos, this ability of rate adaptation may also be needed even when the (aggregate) bandwidth of the VPN pipe is guaranteed. The rest of this paper is devoted to addressing these issues. In Section 3 we develop a slightly modified version of TCP—the *controlled* TCP—for transmitting the segments of the reliable flow of a video in a manner that controls the bandwidth sharing of the reliable flow with the unreliable flow. In Section 4 we design a simple rate adaptation scheme for the unreliable flow that takes the network condition as well as the application information into consideration. We also propose segment-level rate control schemes that deal with relatively "long-term bandwidth drought."

3 Controlled Transmission of Reliable Flows

In this section we present the controlled TCP (cTCP) scheme for transmission of the reliable flow of a video from a central server to a proxy server across a best-effort wide-area backbone network. A major objective in designing the cTCP is to attain some controllability and predictability in the overall system performance. In particular, we want to exert some degree of control on the interaction of the reliable and unreliable flows of various video streams sharing the same VPN pipe, in order to provide consistent and controlled video quality assurance to end users. In the following, we first motivate the design of the controlled TCP scheme, then describe the scheme in detail. The performance of cTCP is evaluated using simulations.

3.1 The Controlled TCP Scheme

Using the staggered two-flow video streaming technique, the reliable flow of a video is transmitted one segment *ahead of time*, i.e., the **I** frames of the segment being transmitted are not needed immediately. (They only need to be streamed to an end user in the next segment period.) Given that the segment size is in the order of minutes, this allows for sufficient time to recover any lost packets during the transmission of the **I** frame segment across the best-effort network. This is in contrast to the transmission of the segments of the unreliable flow, which are delivered in *real time*. In other words, the **P**/**B** frames of the current segment of the unreliable flow are

transmitted on demand, as they are needed by the proxy server[5]. Due to their time constraints, lost packets of the **P**/**B** frames may not be recovered in time via retransmission.

Instead of reinventing the wheel, we decided to use the TCP transport protocol to reliably transmit each segment of the reliable flow, however with some modification to allow for application level throughput control. *Note that if sufficient bandwidth along the VPN pipe is available*, each segment of the reliable flow can be delivered across the best-effort wide-area backbone before its deadline, i.e., the end of the segment. However, directly applying TCP for transmitting the segments of the reliable flow has some *undesirable* effects on the unreliable flow. Recall that TCP employs a *slow start* mechanism to explore available network bandwidth in the beginning, and uses the additive increase and multiplicative decrease (AIMD) algorithm for flow/congestion control. In other words, a TCP flow will attempt to maximize its throughput by injecting as many packets as the available network bandwidth allows, and enters into a steady state that oscillates with periodic packet losses. In the context of the staggered two-flow video streaming technique, this "greedy" behavior of TCP unfortunately has an adverse effect on the unreliable flow: when the available bandwidth is sufficient for transmitting a segment of the reliable flow during a segment period, TCP will grab more bandwidth than what it needs, transmitting the reliable segment "in a blast." The unreliable flow suffers more packet losses as a result of its bandwidth competition with the reliable flow. This problem is further compounded when multiple video streams share the same VPN pipe. Besides, the bandwidth requirements of the reliable flows of different video streams may be different. However, using TCP, these flows will share the available bandwidth more or less equally. As new video streams are initiated or existing video streams terminate, the bandwidth shares of the flows also change, independent of their actual bandwidth requirements.

To address these issues, we develop the controlled TCP (cTCP) scheme to control the bandwidth sharing among the reliable flows of various videos, and to minimize the impact of reliable flows on the unreliable flows. Using cTCP, we are able to provide more predictable overall system performance and offer controlled video quality assurance to end users. We show that this scheme is effective when the (aggregate) bandwidth of the VPN pipe is larger than the total bandwidth requirement of the video streams currently being transmitted. At the end of this section we will briefly discuss how we may ensure this condition to hold by performing application-level admission control at either the proxy server or the central server. In Section 4 we will further discuss how to deal with the situation when this condition is not met.

The basic idea behind the controlled TCP scheme is the realization that in delivering each segment of the reliable flow of a video from the central server to the proxy server, only the amount of bandwidth that is sufficient to transmit the segment before its deadline is needed. More bandwidth for the reliable flow is not necessary, and may even be harmful to the unreliable flow. Hence when there is sufficient network bandwidth, we should limit the TCP throughput of a segment transmission to what is needed. This leads to the concept of the *target throughput* of a segment: Given a segment of length T (measured in seconds), let S be the amount of the data

[5] In general a small start-up delay can be introduced so that the proxy server can smooth its transmission of the unreliable flow using its buffer space [19]. This, in particular, can be done when a small initial prefix of the first segment of the unreliable flow is cached/pre-stored at the proxy server [16]. In our simulation study we assume this is the case.

```
    At the end of each adjustment interval
1.    Obtain the smoothed RTT for the past interval.
2.    Obtain the controlled TCP packet loss information.
3.    if (there are TCP packet losses)
4.      W_target = (T_cTCP · RTT)/(0.75 · MSS)
5.    else
6.      W_target = (T_cTCP · RTT)/MSS
```

Figure 3: The cTCP target window adjustment algorithm.

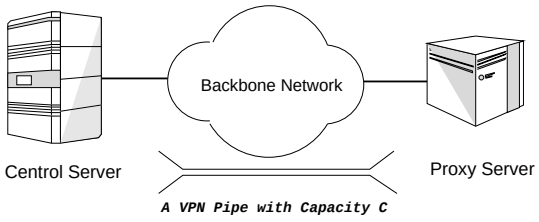


Figure 4: Simulation setting.

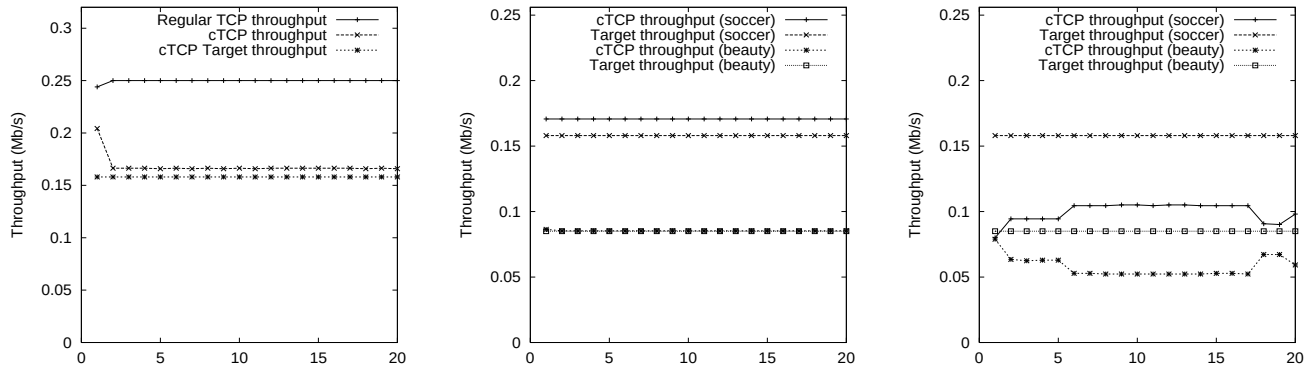
contained in the segment (measured in bits). Then its target throughput, denoted by *target* T_{cTCP}, is given by $\frac{S}{T(1-\hat{p})}$, where $\hat{p}$ is the cTCP retransmission threshold, e.g., 0.05. The factor $1/(1-\hat{p})$ accounts for the potential bandwidth consumed by retransmissions in a network with a packet loss rate of at most $\hat{p}$. Recall that TCP uses a window-based flow/congestion control mechanism, where the number of outstanding packets that can be injected into the network is limited by a window $W = min(W_{cwnd}, W_{recv})$, where W_{cwnd} is the congestion window size, and W_{recv} is the receiver window size. To control the throughput of TCP, we replace[6] W_{recv} by a *target window* W_{target}. Hence when $W_{target} < W_{cwnd}$, the throughput of TCP is limited by W_{target}, even though more packets can be injected into the network without causing congestion. (When $W_{target} \geq W_{cwnd}$, the throughput of TCP is determined by the congestion window W_{cwnd}, as is in the regular TCP.)

The question now is how to determine W_{target} from a given *target* T_{cTCP}. To this end, we use a simple result from the TCP modeling work [6, 10, 11, 13]. It is well known that when there are (small) packet losses, then the (steady state) TCP throughput is given approximately by the simple formula $0.75 \cdot W \cdot MSS/RTT$, where MSS is the TCP maximal segment size, and RTT is the (smoothed) round trip time. When there is no packets lost, then the TCP throughput is roughly $W \cdot MSS/RTT$. Given these formulas, we can compute W_{target} from a given *target* T_{cTCP} as follows: $W_{target} = (T_{cTCP} \cdot RTT)/(0.75 \cdot MSS)$, if there are packet losses; $W_{target} = (T_{cTCP} \cdot RTT)/MSS$, if there is no packets lost. Note that in using the above formulas to compute W_{target}, we need to measure RTT, the round trip time. Depending on the measured RTT, W_{target} may vary over time. To take possible changes of RTT into account, we adjust W_{target} periodically after each *adjustment interval*, during the transmission of a segment of the reliable flow. Compared to RTT, the adjustment interval is relatively larger (in a few seconds), yielding more stable evolution of W_{target}. The adjustment algorithm used in our controlled TCP scheme is shown in Figure 3. At the end of each adjustment interval, we adjust W_{target} using the (smoothed) RTT measured during the interval. Depending on whether there are packet losses or not during the interval, one of the two formulas is used to compute W_{target}.

3.2 Performance Evaluation of cTCP

We now evaluate the performance of the controlled TCP (cTCP) scheme using simulations, and compare it with the regular TCP scheme. For convenience, throughout this subsection we refer to a reliable flow that is

[6] Note that since the receiver of the reliable flow is the proxy server, which is assumed to have sufficient receiving buffer space to accommodate the transmission of the reliable flow.



(a) Single cTCP or TCP flow. (b) At target throughputs. (c) Sharing limited resources.

Figure 5: Throughputs of cTCP flows under different situations.

transmitted using cTCP as a cTCP flow, and a reliable flow that is transmitted using (regular) TCP as a TCP flow, an unreliable flow that is transferred using RTP/UDP as an UDP flow. A video stream whose reliable flow is transmitted using cTCP and unreliable flow is transferred using RTP/UDP is referred to as a cTCP/UDP video session, and a video stream whose reliable flow is transmitted using TCP and unreliable flow is transferred using RTP/UDP is referred to as a TCP/UDP video session. Through this comparative simulation study, we will demonstrate that the cTCP scheme indeed provides us with some ability to control the bandwidth sharing among various reliable flows and minimize their impact on the performance of unreliable flows. As a result, it yields more *stable* and *predictable* performance that is critical in providing controlled video quality assurance to end users. This performance predictability also enables us to perform a form of "application-level admission control" at the central server/proxy server, as we will briefly discuss at the end of this section.

The simulation setting is shown in Figure 4, where the central server and the proxy server are connected by a VPN pipe with a bottleneck capacity of C. The actual value of C depends on the specific simulation scenario. The buffer size of the bottleneck link is determined based on the link capacity in such a manner that the maximum queueing delay is less than 50 ms. The propagation delay of the VPN pipe is set to 40 ms. The TCP maximum segment size (or the network MTU) of the VPN pipe is set to be 1500 bytes. All data packets are assumed to be the same size, with a payload of 1400 bytes. The W_{target} adjustment interval used in the controlled TCP scheme is 8 seconds.

In the first set of simulations, we investigate the effectiveness of using W_{target} in controlling the throughput of a cTCP flow. Hence in this set of simulations we consider only reliable flows. The bottleneck link capacity C is set to be 0.256Mb/s. Figure 5(a) shows the results where the VPN pipe is used to transmit the reliable flow of a video *Soccer* using either cTCP or regular TCP. The reliable flow has a target throughput of 0.158Mb/s. From the figure we see that the cTCP flow attains a stable throughput close to its target throughput, whereas the TCP flow grabs all the available bandwidth, attaining a throughput approximately equal to the bottleneck link capacity.

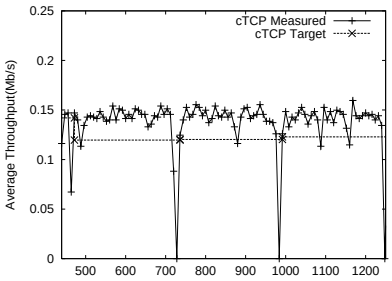


Figure 6: cTCP Throughputs in transmission.

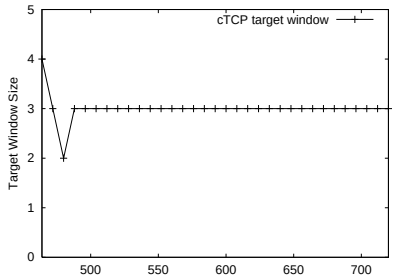
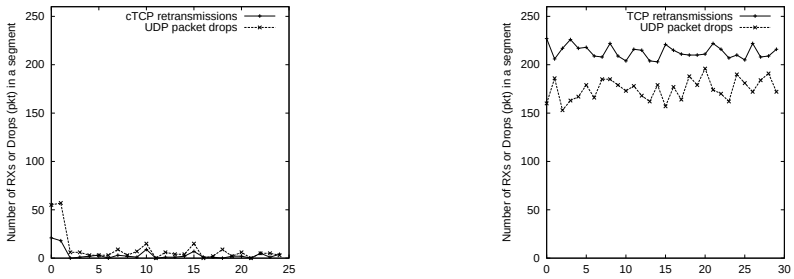


Figure 7: The evolution of the cTCP target window during the transmission of the third **I** frame segment.

(Note that the x-axis in the plots of Figure 5 represents time, indexed by the adjustment interval numbers, counting from 0. Hence the interval no. 20 represents an elapsed time of $20 \cdot 8 = 160$ seconds.) Figure 5(b) shows the results where two reliable flows transmitted using cTCP share the VPN pipe, with target throughputs of 0.158Mb/s (from video *Soccer*) and 0.085Mb/s (from video *Beauty and Beast*), respectively. The sum of the two target throughputs is less than the bottleneck link capacity. In this case, we see that each cTCP flow attains a stable throughput which is close to its target throughput. This is opposed to the situation when the flows are transmitted using regular TCP: the bottleneck link capacity is shared equally between both flows, regardless of their target throughputs (the corresponding plot is not shown here.) To illustrate what happens when the sum of the target throughputs of cTCP flows is greater than the bottleneck link capacity, we run another simulation, where the same two cTCP flows (from *Soccer* and *Beauty and Beast*) share the VPN pipe, but the bottleneck link capacity of the VPN pipe is reduced to 0.22 Mb/s, which is less than the sum of the target throughputs of the two cTCP flows, 0.243 Mb/s. The result is shown in Figure 5(c). We see that the cTCP flow with the higher target throughput (*Soccer*) obtains more bandwidth than the cTCP flow with the lower target throughput *(Beauty and Beast).* In this case cTCP still has some effect on the bandwidth shares of the two flows.

From the above results, we see that when the bottleneck link capacity of the VPN pipe is larger than the total target bandwidth requirement of all the reliable flows, the controlled TCP scheme is effective in controlling the bandwidth sharing among these flows. We now turn our attention to the interaction between reliable flows and unreliable flows, in particular, the impact of reliable flows on packet losses experienced by unreliable flows. In this set of simulations we assume that *the aggregate bandwidth of the VPN pipe is sufficient to satisfy the total bandwidth requirement of all the video streams (including the unreliable flows) currently sharing the VPN pipe.* As mentioned earlier, at the end of this section we will briefly discuss how an application-level admission control at the central server/proxy server may be used to ensure that this condition holds. In the next section we will present rate adaptation and control algorithms to deal with the situation where this condition does not hold.

We first show the results of delivering a single video stream across the VPN pipe using the staggered two-flow video streaming technique. The video trace used in this simulation is an approximately 100 minute long sequence from the MPEG-1 encoded *Star Wars,* with a frame rate of 24 frames/s and a GOP pattern of 12 frames (**IBBPBBPBBPBB**). The video stream is divided into a reliable flow (containing 128000 **I** frames) and



(a) cTCP/UDP session

(b) TCP/UDP session

Figure 8: The number of packet retransmissions and drops.

an unreliable flow (containing 1408000 **P**/**B** frames). Both flows are partitioned into segments with a length of 256 seconds. The *total* average bandwidth requirement of the two flows together is 0.495Mb/s, the total maximum bandwidth requirement is 0.529 Mb/s. In this simulation we assume that the bottleneck link capacity is set to 0.529 Mb/s, i.e., equal to the total maximum bandwidth requirement of the two flows. We also assume that the unreliable flow starts 8 seconds (an adjustment interval) later than the reliable flow *allowing cTCP to obtain the initial RTT measurement and set* W_{target}. (This delay in starting the UDP flow can be masked by, for example, caching an initial prefix of the unreliable flow at the proxy server.)

Figure 6 shows the *measured* average throughput (the y-axis) of the cTCP flow during the transmission of the third, fourth, and fifth segments of the video. From the results we see that the cTCP flow meets the delivery deadline of each segment and attains a stable measured throughput close to the target throughput of each segment. In particular, the third, fourth, and fifth **I** segments are delivered, respectively, by the 720th, 967th, and 1248th second, all ahead of their respective deadlines (the 768th, 1024th, and 1280th second). Note the measured cTCP throughput dips at the end of each segment because the transmission of the segment is completed. The **P**/**B** segments of the UDP flow are also delivered smoothly with only a few packet losses (as we will see shortly).

Figure 8(a) shows the numbers of cTCP packet retransmissions as well as that of UDP packet losses during the transmission of the entire video session (the x-axis is indexed by the segment numbers.) We see that the cTCP flow only experiences a small number of packet retransmissions in each segment. The cTCP flow reaches a steady state with a stable W_{target}. (See Figure 7.) After that, the cTCP flow injects packets into the network at a steady pace. Because the cTCP flow grabs only the bandwidth it needs, the corresponding UDP flow obtains sufficient bandwidth for its packet transmission. As a result, it experiences very a few packet losses. This is in contrast with the scenario where the reliable flow is transmitted using regular TCP, as is shown in Figure 8(b). In the scenario, due to the "greediness" of TCP, both the TCP flow and UDP flow experience relatively large number of losses periodically during the transmission of each segment.

To further demonstrate the advantage of cTCP over regular TCP, we look at the packet losses experienced by both reliable flows and unreliable flows, when multiple video sessions of *Star Wars* are transmitted over a

	Number of Flows	1	5	10	20	50
TCP Retransmissions	cTCP/UDP Session	894	15612	19091	21548	63053
	TCP/UDP Session	9994	62344	107836	184110	555509
UDP Packet Drops	cTCP/UDP Session	709	11851	23594	51416	64709
	TCP/UDP Session	3463	28432	75283	211892	349564
Total	cTCP/UDP Session	1603	27463	42685	72964	127762
	TCP/UDP Session	13457	90776	183119	396002	905073
$\frac{Total(cTCP/UDPSession)}{Total(TCP/UDPSession)}$		12%	30%	23 %	18%	14%

Table 3: The number of TCP retransmissions and UDP packet losses.

VPN pipe. In this set of simulations, the bottleneck link capacity is set to be the same as the total maximum bandwidth required by the video sessions, and each video session starts randomly within a short period of time. The two rows of Table 3 show the number of cTCP/TCP packet retransmissions as well as the number of UDP packet drops under different numbers of video sessions, where the reliable flows are transmitted using either cTCP or regular TCP. The last row shows the percentage of the total packet retransmissions and drops in cTCP/UDP sessions over the ones in TCP/UDP sessions. In general, cTCP/UDP sessions experience significantly fewer packet drops and retransmissions. Because of the controlled bandwidth sharing of the cTCP flows, there is always sufficient bandwidth for the UDP flows to transmit their packets. Addition simulations (results not shown here, due to space limitation) we conducted show that, when the bottleneck link capacity of the VPN pipe is slightly higher than (e.g., 1.1 times or more) the total maximum video rate requirement, cTCP/UDP sessions can achieve no packet drops and retransmissions, while TCP/UDP sessions still have a large number of packet drops and retransmissions. These observations also hold in simulation results we obtained using other video traces.

We now investigate the potential impact of UDP packet drops on the user perceived video quality. For this purpose, we introduce a few metrics. For a given video session, let E_f denote the number of **P**/**B** frames that are affected by packet losses (i.e., at least one packet of the frame is lost during the transmission), [7] and E_g is the number of GOPs that have at least one affected **P**/**B** frame. Moreover, we use G_{avg} and G_{max} to denote, respectively, the average number of affected **P**/**B** frames per affected GOP, and the maximum number of affected **P**/**B** frames per affected GOP, computed among all affected GOPs. Similarly, we use G^{con}_{avg} and G^{con}_{max} to denote, respectively, the average number of *consecutive* affected **P**/**B** frames per affected GOP, and the maximum number of *consecutive* affected **P**/**B** frames per affected GOP, again computed among all affected GOPs. The metric G^{con}_{total} denotes the total number of *consecutive* affected **P**/**B** frames across all affected GOPs. The values of these metrics for the regular TCP/UDP video sessions are shown in Table 4, where the results are obtained by averaging over all the TCP/UDP sessions. In contrast, the corresponding results of the cTCP/UDP video sessions are shown in Table 5. Clearly, cTCP/UDP sessions outperform TCP/UDP sessions in each category, especially in the total number of consecutive affected frames and the maximum number of consecutive affected frames which are the causes of the worst damage to playback quality. *Again, given a little higher bottleneck capacity on the VPN pipe, cTCP/UDP sessions can achieve no packet drops or retransmissions. All the entries in Table 5 will be zero.*

[7] Since a **P**/**B** frame is usually small, a packet loss basically destroys the frame.

n	E_f	E_g	G_{max}	G_{avg}	G^{con}_{total}	G^{con}_{max}	G^{con}_{avg}
1	2862	2365	5	1.2	139	3	2.0
5	4856	3799	5	1.3	282	3	2.1
10	6501	4605	5	1.4	519	4	2.1
20	8971	5594	7	1.6	848	5	2.1
50	6196	4530	6	1.4	379	5	2.1

Table 4: Affected **P**/**B** frames of the UDP flow in a TCP/UDP session.

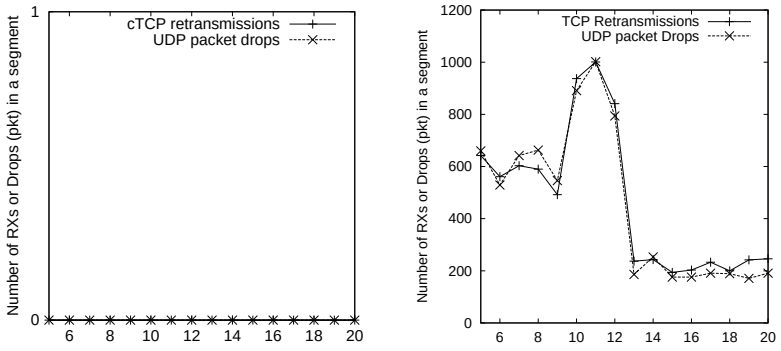
n	E_f	E_g	G_{max}	G_{avg}	G^{con}_{total}	G^{con}_{max}	G^{con}_{avg}
1	617	541	2	1.1	9	2	2.0
5	2078	1841	4	1.1	69	4	2.0
10	2130	1867	3	1.1	62	3	2.0
20	2272	1988	4	1.1	63	3	2.0
50	1134	1080	3	1.1	8	2	2.0

Table 5: Affected **P**/**B** frames of the UDP flow in a cTCP/UDP session.

Under the same conditions, TCP/UDP sessions will have similar behavior as shown in Table 4.

In the final set of simulations we show the impact of dynamic video session arrival and departures on the system performance. In this set of simulations we set the bottleneck link capacity of the VPN pipe to 1.1 times of what is needed to carry five concurrent video sessions of *Star Wars*. At the beginning, we have four on-going video sessions. Then at segment no. 10 (i.e., after 2560 seconds), a new video session joins the four on-going video sessions. At segment 13 (i.e., after 3328 seconds), two video sessions terminate. Figures 9 (a) and (b) show the impact of the video session arrival and departures on the packet losses experienced by one of the on-going video sessions, where in (a) the reliable flows of the video sessions are transmitted using cTCP, while in (b) the reliable flows of the video sessions are transmitted using regular TCP. We see that the dynamic session arrival and departures have no visible impact on the on-going cTCP/UDP sessions in this case. In contrast, the dynamic session arrival and departures have a strong impact on both the number of packet retransmissions of the TCP flow and the packet drops of the UDP flow in a TCP/UDP session. This is because regular TCP always attempts to distribute the bandwidth equally among the reliable flows of the current on-going video sessions. When a new video session joins or an existing session leaves, the bandwidth has to be redistributed among the TCP flows. This causes fluctuation in the packet losses experienced by the video sessions, which in turn will induce fluctuation in the video quality perceived by end users. Using controlled TCP, we are able to avoid this fluctuation in the bandwidth seen by the video sessions, therefore providing more consistent video quality to end users.

Before we leave this section, we briefly comment on the performance predictability offered by the controlled TCP scheme, which may be employed to perform a form of *application-level admission control* at either the central video server or the proxy server. This application-level admission control can be used to ensure that the total bandwidth requirement of all the video sessions carried by the VPN pipe is less than the bottleneck link capacity of the VPN pipe. Suppose that the (aggregate) bandwidth of the VPN pipe is guaranteed and known (e.g., via some service level agreement with the network service provider.) The central server (or proxy server) will only allow a new video session to be carried over the VPN *if* its maximum bandwidth requirement (of both the



(a) cTCP/UDP Session

(b) TCP/UDP session

Figure 9: Fluctuations of the arrival and departures of video sessions.

reliable and unreliable flows) is less than the "residual" bandwidth on the VPN. Here the leftover bandwidth is the difference between the total bandwidth of the VPN pipe and the total bandwidth requirement of the on-going video sessions. In the case where the (aggregate) bandwidth of the VPN pipe is not guaranteed or known, we can resort to some measurement-based techniques. For example, we can use the measured throughputs of both cTCP flows and UDP flows as well as the measured packet losses of these flows as a criteria to determine whether a new video session can be admitted. If their measured throughputs of the flows are close to their corresponding target bandwidth requirements, *and* the measured packet losses are significantly below certain pre-set packet loss thresholds for both cTCP flows and UDP flows (e.g., 0.05), the new video session is then admitted. Otherwise it is rejected. We are currently carrying out a more in-depth study on this subject. In the next session we propose rate adaptation schemes to deal with the situation where there is *no* sufficient bandwidth to carry all the on-going video sessions, due to, for example, the bandwidth fluctuation in the VPN pipe.

4 Rate Adaptation and Segment Control

When the (aggregate) bandwidth of the VPN pipe is not sufficient to carry the currently on-going video sessions, both the reliable flows and unreliable flows of these video sessions may experience a large number of packet losses. Such packet losses not only degrade the user perceived video quality due to the affected **P** and **B** frames, but also cause cTCP to reduce the transmission rates of the reliable flows below their target throughputs. In this section we discuss how this problem can be addressed by performing rate adaptation and control at two levels: *rate adaptation of unreliable flows within segments to deal with short-term transient network congestion, and rate control at segment-level for handling relatively long-term persistent network congestion.* The proposed schemes are described below.

```
1.  if ( measured T_cTCP < target T_cTCP)
2.    the UDP flow backs off (target T_cTCP - measured T_cTCP) by discarding B/P frames.
3.  else
4.    if ( measured T_cTCP ≃ target T_cTCP)
5.      if (the number of cTCP retransmissions > the retransmission threshold)
6.        the UDP flow backs off by discarding one or more B/P frames.
7.      else
8.        if (the loss rate of the UDP flow > its loss rate threshold)
9.          the UDP flow backs off by discarding one or more B/P frames.
10.   if (No cTCP retransmissions and No UDP packet losses)
11.     the UDP flow increases by adding one or more B/P frames.
12.   else
13.     No operations.
```

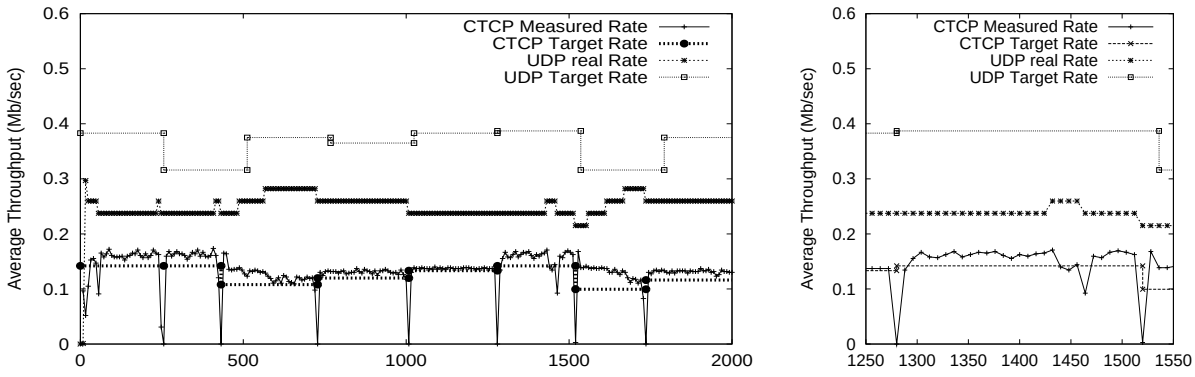
Figure 10: Rate adaptation algorithm for unreliable flows.

4.1 Rate Adaptation of Unreliable Flows

When a video session experiences network congestion, as indicated by considerable amount of packet losses seen by either the reliable flow or the unreliable flow, we reduce the transmission rate of the unreliable flow by selectively discarding certain **B/P** frames. In doing so, we attempt to achieve two objectives: 1) By reducing the transmission rate of the unreliable flow whenever network congestion is detected, we cede more bandwidth to the reliable flow so that the target throughput of the current segment of the reliable flow can be maintained. 2) In the face of network congestion, we want the user perceived video quality to degrade gracefully in a *controlled* manner by selectively discarding the least important **B/P** frames to minimize packet losses.

A high-level description of the rate adaptation algorithm used in our staggered two-flow video streaming technique is shown in Figure 10. Using this algorithm, a video session keeps monitoring the network status through TCP ACKs from its reliable flow and periodical packet loss reports (via RTP receiver report) from its unreliable flow. Based on the target throughput, the measured throughput of the reliable flow, the observed cTCP retransmissions and UDP packet losses, the video session adjusts the transmission rate of its unreliable flow at the end of each adjustment interval. If the measured throughput of the reliable flow is considerably less than its target throughput (line 1 in Figure 10), the transmission rate of the unreliable flow is reduced by the amount of the difference between the target throughput and measured throughput of the reliable flow. If the measured throughput of the reliable flow is approximately equal to its target throughput, but if the number of cTCP retransmissions is larger than a retransmission threshold (e.g., a packet loss rate of 0.05), we reduce the transmission rate of the unreliable flow by a small amount through discarding one or more **B/P** frames (lines 6). Furthermore, if the UDP packet loss rate of the unreliable flow exceeds a loss rate threshold (e.g., 0.05), we reduce the transmission rate of the unreliable flow by a small amount by discarding one or more **B/P** frames (lines 9). However, if neither the reliable flow nor the unreliable flow experiences any packet losses and the target throughput of the reliable flow is met, we will increase the transmission rate of the unreliable flow (if its rate has been previously reduced). In other cases, no rate adjustment is performed.

In reducing the transmission rate of the unreliable flow, we employ a simple selective frame discard scheme [25]. We select frames to be discarded on a per-GOP basis, and always choose the least important frames of a GOP to discard. This means that the **B** frames will always be considered before the **P** frames. Among the **B** frames, the ones with the closest deadlines are always chosen first. Among the **P** frames, the ones that no other **P** frames are dependent on are chosen first.



(a) The first 2000 seconds.

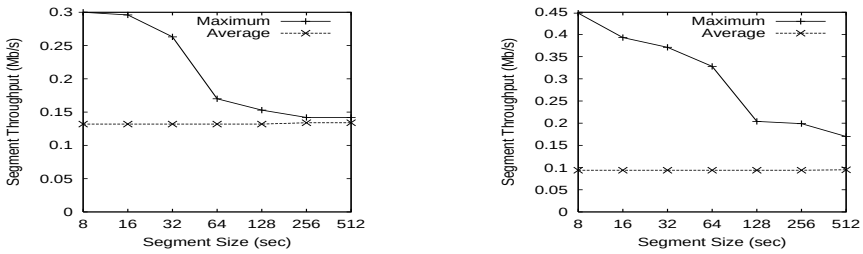
(b) Segment 6.

Figure 11: Rate adaptation of unreliable flow.

We now show the performance of the above rate adaptation scheme through a simple simulation. A cTCP/UDP video session *Star Wars* is carried over a VPN pipe, whose bottleneck capacity is set to 0.4 Mb/s. The average bandwidth requirement (for both the reliable flow and the unreliable flow) of the *Star Wars* video session is 0.495 Mb/s, larger than the bottleneck capacity of the VPN pipe. The average rate of the reliable flow is only 0.134 Mb/s, considerably smaller than the bottleneck capacity of the VPN pipe. Figure 11(a) shows the measured throughput/rate of both the reliable flow and the unreliable flow as well as the target throughput/rate of both flows for the first 2000 seconds of the video transmission. We see that because the unreliable flow backs off its transmission rate from its target rate, the reliable flow is able to attain its target throughput. Note that the dips in measured cTCP throughput are due to the end of **I** segment transmission. Figure 11(b) shows the rate adaptation process in a smaller time scale—during the period in which the 6th **I** segment of the reliable flow (from the 1288th second to the 1520th second) is delivered. This figure clearly demonstrates the effectiveness of the rate adaptation of the unreliable flow in assisting the reliable flow to attain its target throughput. Other simulations using different video traces give us the similar results.

4.2 Segment Control

Rate adaptation of the unreliable flows by selective frame discard is most effective in dealing with less severe network congestion, in particular, short-term transient network congestion. However, when the network congestion persists, in particular, when the average bandwidth of the VPN pipe over a segment period is lower than the



(a) *Star War* (b) *Beauty and Beast*

Figure 12: **I**-Segment target throughput v.s. segment size.

total target throughputs of the reliable flows of the on-going video sessions, simply reducing the transmission rates of the unreliable flows is no longer sufficient. First, further discarding the **B**/**P** frames may significantly degrades the user perceived video quality. Second, and more importantly, we are also likely to risk not delivering the **I** frame segment in time, which in turn will significantly degrade the user perceived video quality. In such situations, more drastic rate adaptation is needed. To this end, we resort to rate control schemes at segment level. Depending on the severity of the bandwidth drought we experience on the VPN pipe, two segment-level rate control mechanisms can be employed.

The first mechanism relies on the inherent burstiness of video streams. Figures 12 (a) and (b) show the maximum as well as the average throughput requirements of the **I** frame segments of two video traces *Star Wars* and *Beauty and Beast*, as the function of segment size (measured in seconds). We see that although the average throughput requirement is independent of the segment size , the maximum throughput requirement decreases as the segment size increases. So far in our presentation of the staggered two-flow video streaming technique, we have assumed that the segment sizes of all the segments are fixed and the same. However, this need not be the case. The size of the first **I** frame segment is mostly determined by the amount of disk space at the proxy server and the number of videos whose first **I** frame segments are staged (pre-stored) at the proxy server. Therefore, the size of the first **I** frame segment must be determined *a priori*. However, the **I** frame segments after the first segment are only going to be cached at the proxy server. Hence, depending on the number of the on-going video sessions and the amount of disk space at the proxy server for video caching, the segment size can be dynamically determined at run time. The first segment-level rate adaptation scheme is designed based on these observations. In the face of severe network congestion, we dynamically increase the segment size to reduce the target throughput of the future **I** frame segments of the currently on-going video sessions. Furthermore, instead of starting transmission of the next **I** segment only at the beginning of the next segment period, we start the transmission of the next **I** segment immediately after the finish of the previous **I** frame segment, whenever more disk cache space is available at the proxy server. (This has been used in the simulations in Section 3.)

Clearly, the above segment-level rate control mechanism works only when the bandwidth deficit in the VPN

pipe can be accommodated by increasing the segment size. When this is not the case, the only way to circumvent the network congestion problem is to reduce the throughput requirement of the future **I** frame segments considerably. To do so, we encode a video in multiple quantization levels. When severe long term network congestion occurs, we dynamically switch to a **I** frame segment that is encoded using a lower quantization level and has a much lower target throughput requirement. The corresponding **P**/**B** segment is also switched. Performance study of these segment-level rate control mechanisms together with the application-level admission control algorithm will be presented in a future paper.

5 Conclusions and Future Work

In this paper we have proposed and developed a novel proxy-assisted video streaming technique—the staggered two-flow video streaming technique—for delivering videos across a best-effort wide-area backbone network from a central server to a video proxy server. The objective of the proposed video streaming technique is to provide controlled video quality assurance by intelligently taking advantage of the disk space at the proxy server, while circumventing the disk I/O bandwidth limitation at the proxy server in the mean time. Our technique is designed for stored videos that use compression schemes with inter-frame dependency, such as MPEG. Using the proposed staggered two-flow video streaming technique, we divide each video stream in two separate substreams and deliver them using different mechanisms: a portion of the video (containing the essential **I** frames) is delivered reliably ahead of time and cached at the proxy server, while the rest of the video (containing the less essential **P** and **B** frames) are delivered unreliably in real-time.

We designed a slightly modified version of TCP with application throughput control, the *controlled TCP (cTCP)*, for reliable and timely delivery of the **I** frame segments. Given that the (aggregate) bandwidth of the VPN pipe is sufficient, the controlled TCP is shown to be capable of controlling the bandwidth sharing between reliable flows and unreliable flows, thereby minimizing the impact of the reliable flow transmission on packet losses experienced by the unreliable, real-time RTP/UDP flows. As a result, our video streaming technique yields more *stable* and *predictable* performance that is critical in providing consistent and controlled video quality assurance to end users. To deal with both transient and persistent network congestion, we have also developed rate adaptation schemes at two levels: rate adaptation of the unreliable flows within segments, and segment-level rate control through dynamic segment size adjustment and dynamic switching between video copies encoded at different quantization levels. Both rate adaptation and control schemes take into account the application-specific information such as bandwidth requirement and frame types. Through extensive simulations, we have illustrated that, by intelligently taking advantage of the disk space at the proxy server, the staggered two-flow video streaming technique can indeed provide controlled video quality assurance, while circumventing the disk I/O bandwidth limitation at the proxy server. We are currently conducting further performance study, in particular, evaluation of the proposed segment-level rate control mechanisms in conjunction with the application-level admission control algorithm outlined in the paper. The initial implementation of the proposed staggered two-flow streaming video technique is under way. Experimentation over the Internet II backbone will be carried out in the near future.

References

[1] Akamai Technologies, http://www.akamai.com.

[2] E. Amir, S. McCanne, and H. Zhang, "An Application Level Video Gateway," in Proc. ACM Multimedia'95, Nov, 1995.

[3] N. Cardwell, S. Savage, and T. Anderson, "Modeling TCP Latency," in Proc. IEEE Infocom'00, Tel-Aviv, Israel, March, 2000.

[4] D. Clark and D. Tennenhouse, "Architectural Considerations for a New Generation of Protocols," in Proc. SIGCOMM'90.

[5] W. Feng, M. Liu, B. Krishnaswami, and A. Prabhudev, "A Priority-Based Technique for the Delivery of Stored Video Across Best-Effort Networks," in Proc IS&T/SPIE Multimedia Computing and Networking 1999, San Jose, CA, Jan. 1999.

[6] S. Floyd, M. handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," in Proc. SIGCOMM'00.

[7] S. Floyd, and K. Fall, "Promoting the Use of End-to-End Congestion Control in the Internet," IEEE/ACM Transactions on Networking, August 1999.

[8] Fast Forward Networks, http://www.ffnet.com.

[9] V. Jacobson, and M. J. Karels, "Congestion Avoidance and Control," in Proc. SIGCOMM'88.

[10] J. Mahdavi and S. Floyd, "TCP-friendly Unicast Rate-based Flow Control," http://www.psc.edu/networking /papers/tcp_friendly.html.

[11] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The Macroscopic Behavior of the TCP Congestion Avoidance Algorithms," Computer Communication Review, Vol.27(3), July 1997. pp.67-82.

[12] J. M. McManus and K. W. Ross, "Video-on-demand over ATM: Constant-rate Transmission and Transport," IEEE J. Selected Areas in Communications, Vol.14(6), pp.1087-1098, Aug. 1996.

[13] J. Padhye, J. Kurose, D. Towsley, and R. Koodli, "A Model Based TCP-Friendly Rate Control Protocol," in Proc. IEEE NOSSDAV'99 (Basking Ridge, NJ, June 1999).

[14] R. Rejaie, H. Yu, M. Handely, and D. Estrin, "Multimedia Proxy Caching Mechanism for Quality Adaptive Streaming Applications in the Internet," Technical report 99-709, Computer Science Department, USC.

[15] O. Rose, "Statistical Properties of MPEG Video Traffic and Their Impact on Traffic Modeling in ATM Network," TR-101, Institute of Computer Science, University of Wurzburg, Germany, Feb. 1995.

[16] S. Sen, J. Rexford, and D. Towsley, "Proxy Prefix Caching for Multimedia Streams," in Proc. IEEE INFOCOM'99.

[17] P. Shenoy, P. Goyal and H. Vin, "Issues in Multimedia Server Design," ACM Computing Surveys, Dec. 1995.

[18] H. Shulzrinne, S. Casner, R. Frederick, and V. Jacboson, "RTP: A Transport Protocol for Real-Time Applications," RFC 1889, 1996.

[19] J. Salehi, Z.-L. Zhang, D. Towsley, and J. Kurose, "Supporting Stored Video: Reducing Rate Variability and End-to-End Resource Requirements Through Optimal Smoothing," in Proc. ACM SIGMETRICS'96.

[20] Real Networks, http://www.real.com.

[21] D. Tennenhouse and D. Wetherall, "Towards an Active Network Architecture," Computer Communication Reviews, Vol.26(2), 1996.

[22] R. Tewari, H. M. Vin, A. Dan, and D. Sitaram, "Resource-based Caching for Web Servers," in Proc. SPIE/ACM Conf. on Multimedia Computing and Networking, Jan, 1998.

[23] Y. Wang, and D. Du, "Weighted Striping in Multimedia Servers," ICMCS'97, Jun. 1997.

[24] Y. Wang, Z.-L. Zhang, D. Du, and D. Su, "A Network Conscious Approach to End-to-End Video Delivery Over Wide Area Networks Using Proxy Servers," In Proc. IEEE INFOCOM'98.

[25] Z.-L. Zhang, S. Nelakuditi, R. Aggarwal, and R. Tsang, "Efficient Selective Frame Discard Algorithms for Stored Video Delivery across Resouce Constrained Networks," in Proc. INFOCOM'99.