STREAMING LOW-DELAY VIDEO OVER AIMD TRANSPORT PROTOCOLS

Ahmed Abd El Al, Tarek Saadawi, Myung Lee
Department of Electrical Engineering, City College and Graduate Center of City University of New York
Convent Avenue at 140th Streey, New York, NY 10031, USA

Keywords: Media Adaptation, Low-Delay Video, SCTP.

Abstract: In this paper, we present adaptation strategies for low-delay video streams over Additive-Increase Multiplicative- Decrease (AIMD) transport protocols, where we switch among several versions of the coded video to match the available network bandwidth accurately, and meet client delay constraints. By monitoring the application buffer at the server, we estimate the current and future server buffer drain delay, and derive the transmission rate to minimize client buffer starvation. We also show that the adaptation accuracy can be significantly improved by a simple scaling to transport protocol send-buffer size. The proposed mechanisms were implemented over Stream Control Transmission Protocol (SCTP) and evaluated through simulation and real Internet traces. Performance results show that the adaptation mechanism is responsive to bandwidth fluctuations, while ensuring that the client buffer does not underflow, and that the quality adaptation is smooth so that the impact on the perceptual quality at the client is minimal.

1 INTRODUCTION

While the majority of traffic on the Internet today is comprised of TCP flows, conventional wisdom holds that TCP is unsuitable for "low-delay" traffic due to its lack of throughput guarantees and insistence on reliability (Krasic, 2001). For these reasons, there have been many proposals for new transport protocols for the purpose of solving the video transport problem over the Internet. These protocols need to be TCP-friendly to ensure that they will not cause network collapse. However, proving a new transport protocol to be TCP-friendly can be difficult, because the dynamics of TCP congestion control is extremely complex (Hsiao, 2000).

AIMD-based transport protocols experience rate variations for two distinct reasons -- the first being the protocol's own congestion control behavior, i.e., the window-based congestion control algorithm, implemented by most of the AIMD-transport protocols, introduces saw-tooth fluctuation in the streaming rate, and the second being competing traffic in the network. Client-side buffers can be used for smoothing out the saw-tooth fluctuation of a flow (Krasic, 2001). However, despite any amount of buffering, competing traffic can have persistent effects on the streaming rate, and consequently on the viewing quality. The problem is more challenging in the case of low-delay video since client buffering is limited by end-to-end latency limit, and also data cannot be prefetched into the client buffer when extra bandwidth is available. Thus streaming video applications must deal with persistent rate changes, before the client-side buffers are overwhelmed. The usual way is to employ quality-adaptation control, adjusting the basic quality-rate trade off of the video.

The primary design goal of quality-adaptation control mechanisms is to adapt the outgoing video stream so that, in times of network congestion, less video data is sent into the network and consequently fewer packets are lost and fewer frames are discarded. This rests on the underlying assumption that the smooth and timely play out of consecutive frames is central to a human observer's perception of video quality. Although a decrease in the video bitrate noticeably produces images of coarser resolution, it is not nearly as detrimental to the

Prepared through collaborative participation in the Communications and Networks Consortium sponsored by the U.S. Army Research Laboratory under the Collaborative Technology Alliance Program, Cooperative Agreement DAAD19-01-2-0011.

Abd El Al A., Saadawi T. and Lee M. (2006).
STREAMING LOW-DELAY VIDEO OVER AIMD TRANSPORT PROTOCOLS.
In *Proceedings of the International Conference on Signal Processing and Multimedia Applications*, pages 50-57
DOI: 10.5220/0001569700500057

perceived video quality as inconsistent, start-stop play out. By switching between different quality levels during the stream, the mechanism makes a fundamental trade-off by increasing the video compression in an effort to preserve a consistent frame rate at the client.

In this paper we focus on sender-driven quality adaptation, for low-delay video streams, to minimize any overheads at the client. In particular, we focus on quality adaptation using stream switching, as it has been shown to provide better viewing quality than adding/dropping layers, due to the layering overhead (Cuetos, 2001). First, we introduce an adaptive stream switching mechanism for low-delay video that does not require either modifications to the network transport protocol at the sender or at the receiver, or support from the network infrastructure. By monitoring the application buffer occupancy, the mechanism detects the network bandwidth variations and estimates the current and future server buffer drain delay, and accordingly it adapts the video transmission rate to minimize the client buffer starvation while ensuring that the adaptation affects the perceptual quality at the client minimally. Then, we show that by scaling the transport protocol *send-buffer* according to the available bandwidth-delay product, the adaptation accuracy can be improved significantly.

Although the presented mechanisms are suitable for video transport over AIMD-based transport protocols in general, we chose to implement them over Stream Control Transmission Protocol (SCTP) (Stewart, 2000). SCTP is an AIMD-based transport protocol, with many attractive features for video transport than TCP. The most attractive features are multi-streaming and multi-homing support. Multi-streaming allows data to be partitioned into multiple streams that have the property of being independently delivered to the application at the receiver. This means that the loss of a data chunk that belongs to a certain stream will only affect the delivery within that stream, without affecting the delivery of other streams. This feature prevents head-of-line blocking problem that can occur in TCP, as TCP supports only a single data stream. Multi-homing allows a single SCTP endpoint to support multiple IP addresses. In its current form, SCTP multi-homing support is only for redundancy. In addition, SCTP provides different reliability levels, which are more suitable for video transport than the strict reliability provided by TCP (Blak, 2002). SCTP congestion control is similar to TCP, thus ensured to be friendly to other TCP flows sharing the same network (Brennan, 2001).

This paper is organized as follows. In section 2, we present related work in video rate and quality adaptation mechanisms. In Section 3, we describe our system architecture. Section 4 describes the adaptation mechanisms and the SCTP *send-buffer* scaling. In Section 5, we evaluate the performance for our proposed mechanisms. Section 6 concludes the paper.

2 RELATED WORK

Multimedia adaptation has been studied for Internet applications, and the adaptive control schemes can be classified into receiver-driven and sender-driven. Receiver driven schemes allow receivers individually to tune the received transmission according to their needs and capabilities. Mehra and Zakhor (Mehra, 2003) modify the TCP protocol at the receiver end to provide video streams a nearly CBR connection over a bandwidth limited access link. Hsiao *et al* (Hsiao, 2001) present Receiver-based Delay Control (RDC) in which receivers delay TCP ACK packets based on router feedback to provide constant bit rate for streaming. While receiver buffers can be used for smoothing out rate fluctuations, buffering is limited by the end-to-end latency limit.

A majority of the sender-driven algorithms may be grouped under quality adaptation schemes. Quality adaptation techniques can further be classified into on-the-fly encoding, adding/dropping layers, and switching among multiple encoded versions. Kanakia *et al* (Kanakia, 1993) estimate the buffer occupancy and the service rate received by the connection at the bottleneck queue through periodic feedback messages from the network. These estimates are used to control the transmission rate of each video frame on-the-fly by adjusting the encoder quantization factor. However, in general, on-the-fly encoding is CPU intensive and thus regarded as unsuitable for streaming real-time video. In the adding/dropping layers scheme, the video stream is partitioned into several layers using scalable coding schemes such as MPEG-4 FGS or interframe wavelet video encoding. Video streaming applications can add or drop enhancement layers to adjust the transmission rate to the available bandwidth. In the switching-versions scheme, the video is encoded at different rates, and therefore different quality levels, and each of these versions is made available to the streaming server as an independent stream. The server detects changes in available bandwidth and switches among the input streams, in order to adapt the transmission rate to the available bandwidth. Quality adaptation that is based on multiple encoded versions has been shown to provide better viewing quality than adding/dropping layers, due to the layering overhead (Cuetos, 2001).

Besides quality adaptation, scheduling algorithms may also be used to improve the multimedia streaming adaptation. Saparilla and Ross (Saparilla, 2000) prefetch future portions of the stored video into the client buffer as bandwidth is available. For low-delay video prefetching is not possible, and the server application transmits the video stream at the consumption rate, unless the available bandwidth is less than the consumption, at which time the stream is transmitted at the available bandwidth rate.

3 SYSTEM ARCHITERCURE

In our architecture, shown in Figure 1, low-delay media is encoded into multiple quality streams, which are fed to the adaptive media server. The server selects one of these streams $R^{in}(t)$ and injects it into the server buffer. The server buffer is drained as fast as the network connection permits, i.e. $R^{out}(t)$. The network output is fed into the client buffer. To smooth out short time scale bandwidth variations and to remove jitter, D units of time of the stream are allowed to build up in the client prefetch buffer before playback begins. The size of this buffer is limited by the maximum end-to-end latency constraints of the system. In addition, without loss of generality, we assume that the streams are CBR encoded.

In this paper, we present adaptation algorithms that enable the server to stream the media across varying network bandwidth conditions while maximizing the video quality at the client.

4 STREAM SWITCHING STRATEGIES

Estimation of network bandwidth in the case of SCTP-based streaming is not straightforward since SCTP hides the network congestion status from the application. A somewhat delayed effect can, however, be seen in the application buffer.

If the server is streaming at a certain rate and the network capacity goes below this rate, this will be reflected as increase in the server buffer occupancy. Similarly, if the server is streaming with a rate lower than the current network capacity, the server buffer will start to empty.

The proposed mechanism monitors the application buffer at the server in order to estimate the current available bandwidth in the network, and accordingly it adapts the streaming rate. It reacts to the network congestion state so as to prevent client buffer underflow (the client prefetch buffer and the server buffer are mirror images of each other), in addition it tries to keep the number of quality changes at the client to a minimum so as to have minimal effect on the user perceived quality. We introduce below the criteria used for stepping up or down the streaming rate. The algorithm is described in the context of a low-delay stream, which the server receives from a live-source and forwards to the client over a SCTP connection.

Consider that we have N available video streams (different encodings or derived sub-streams) with corresponding bit-rates V_j ($j = 1,2,\ldots,N$) and we make the decision to switch at discrete time instances t_k. Follows, we present the strategies to make this decision.

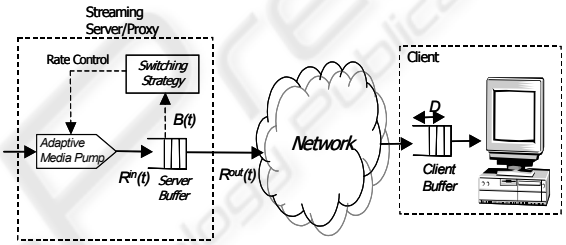

Figure 1: Streaming system architecture.

4.1 Switching Down Strategy

The minimum delay that any incoming packet (at time t_k^+) experiences with the server buffer at measured fullness $B(t_k)$ is the time required to drain the buffer. This delay $\Delta_{k^+}^{\min}$ may be estimated as $\Delta_{k^+}^{\min} = \frac{B(t_k)}{R^{out}(t_k)}$, where $R^{out}(t_k)$ is the output rate estimated at time instant t_k. $R^{out}(t_k)$ can be obtained using a Weighted Exponential Moving Average (WEMA) of the past and current bandwidth observations at the server. In order to satisfy the client delay constraints and prevent the client buffer from underflowing (server buffer from overflowing), we should have $\Delta_{k^+}^{\min} < D$. Hence, whenever we observe $\Delta_{k^+}^{\min} > \alpha D$, we reduce the input rate to the largest available rate smaller than $R^{out}(t_k)$, so that we drain the server buffer build-up and preempt any overflow. The conservative factor α ($0 < \alpha < 1$), is introduced in order to account for possible variations in the input and output rates during sampling

interval $[t_k, t_{k+1})$. Hence, for this interval, we select the input rate $\overline{R}_k^{in}$ as:

$$\overline{R}_k^{in} = \max_{\substack{j=1,..,N \\ V_j < R^{out}(t_k)}} \{V_j\}, \quad (1)$$

where V_j are the N available video bit-rates. The factor α should be selected based on the expected variations in the rates.

This decision strategy aggressively reduces the input rate whenever it estimates buffer drain time as being greater than the computed threshold. Although such strategy follows any reductions in the available output bandwidth rapidly, it does not take into consideration the rate of change of the buffer, thus it can lead to unnecessary reductions in the input rate. For instance, even if the buffer fullness was being steadily reduced (based on a previous switching decision), this strategy could further reduce the input rate. This can lead to under-utilization of available bandwidth, and undesirable quality for the user. Thus we combined the instantaneous decision strategy with a look-ahead strategy that takes into consideration the rate of change of the buffer, through estimating the buffer fullness one sampling interval in the future. The look-ahead strategy estimates the server buffer at sampling instant t_{k+1} as:

$$\hat{B}(t_{k+1}) = \max\left\{\left[B(t_k) + \int_{t_k}^{t_{k+1}} \left(R^{in}(u) - R^{out}(u)\right)du\right], 0\right\}, \quad (2)$$

where $R^{in}(u)$ and $R^{out}(u)$ are the instantaneous input and output rates. If the sampling intervals are chosen small enough, we can assume that the input and output rates are constant over the entire interval. Hence we may rewrite equation (2) as

$$\hat{B}(t_{k+1}) = \max\left\{\left[B(t_k) + \left(\overline{R}_k^{in} - \overline{R}_k^{out}\right)(t_{k+1} - t_k)\right], 0\right\} \quad (3)$$

where $\overline{R}_k^{in}$ and $\overline{R}_k^{out}\left(= R^{out}(t_k)\right)$ are the constant input and output rates interval $[t_k, t_{k+1})$. We then estimate the delay $\Delta^{\min}_{(k+1)^-}$ that would be experienced at the end of this interval (before time instant t_{k+1}) as $\Delta^{\min}_{(k+1)^-} = \frac{\hat{B}(t_{k+1})}{\overline{R}_k^{out}}$. Since during this sampling interval we want to avoid server buffer overflow, we would like to constrain $\Delta^{\min}_{(k+1)^-} \le \beta D$, where β ($0 < \beta < 1$) is another conservative factor introduced to account for possible variations in the input and output rates. Combining with equation (3) we can easily determine that:

$$\overline{R}_k^{in} \le \frac{\beta D \overline{R}_k^{out} - B(t_k)}{(t_{k+1} - t_k)} + \overline{R}_k^{out} \quad (4)$$

We may thus use equation (4) to determine what input rate to switch to such that we avoid server buffer overflow, at the end of the current interval, as a result of the decision. The look-ahead strategy can avoid unnecessarily aggressive reductions and stream switches (thereby improving the visual quality) in the input rate by sometimes borrowing from, and sometimes provisioning for the future. However, it also makes assumptions that the output rate does not change significantly over the interval $[t_k, t_{k+1})$. Hence, when the timescale of network variations is smaller than the sampling interval (i.e. the network conditions change rapidly) the instantaneous decision is likely to outperform the look-ahead decision, and conversely if the sampling interval is smaller than the timescale of network variations, the look-ahead decision is likely to outperform the instantaneous decision.

The combined decision strategy, shown in Figure 2, combines the benefits of these instantaneous and look-ahead decision strategies to minimize the number of stream switches (for better visual quality) while following the available bandwidth accurately, and satisfying the user delay constraints.

4.2 Switching Up Decision Strategy

While we attempt to switch down the server streaming rate as soon as we observe a low network bandwidth, we cannot similarly switch up the streaming rate. This is because switching up too rapidly can actually create congestion in the network and thereby lead to oscillations between switching up and switching down, adversely affecting the video quality. Hence, it might be suitable for the server to attempt to stream at a higher rate, only after a certain duration during which the server does not observe a congestion event. The problem is that there is no explicit signal indicating when the server should switch up. For this reason, our mechanism carries out active experiments by probing the network to ensure that there is enough capacity for the next higher streaming rate. We call these experiments, *switch-experiments*. The *switch experiment* is triggered whenever the server does not experience a congestion event for an interval T_E^i referred to as the *Inter-Experiment timer*.

The experiment is performed by switching to the next higher available streaming rate, such that:

$$\overline{R}_k^{in} = \min_{\substack{j=1,..,N \\ V_j > \overline{R}_{k-1}^{in}}} \{V_j\} \quad (5)$$

1. Compute buffer fullness $B(t_k)$ and estimate $R^{out}(t_k)$
   Set $\overline{R}_k^{in} = \overline{R}_{k-1}^{in}$.
2. Estimate $\Delta_{k^+}^{\min}$ and $\Delta_{(k+1)^-}^{\min}$
3. If $\Delta_{k^+}^{\min} > \alpha D$ and $\Delta_{(k+1)^+}^{\min} > \beta D$
   Determine upper bound on input rate
   $$R_{\max}^{in} = \max\left\{\left(\frac{\beta D\overline{R}_k^{out} - B(t_k)}{(t_{k+1} - t_k)} + \overline{R}_k^{out}\right), \overline{R}_k^{out}\right\}$$
   Select $\overline{R}_k^{in} = \max_{\substack{j=1,\ldots,N \\ V_j < R_{\max}^{in}}} \{V_j\}$
   Else
   Test whether we can switch up
4. Wait until next sampling interval. Goto Step 1.

Figure 2: Combined switching down decision strategy.

and each experiment lasts for a maximum duration of T_S. During the switching experiments, the congestion monitor in the server continues to monitor the network and if no congestion is caused due to the experiment, this is considered an indication that the network can support the next higher bit rate stream. In this case, the server stays at the higher stream. However, if congestion is detected, as indicated in Step 3 of the combined switch down algorithm, the sender reverts to the lower rate. The sender also learns from the failed *switch experiment*, by exponentially backing off the *Inter-Experiment timer* T_E^i for this rate, before retrying the experiment. Backing off the timer is likely to reduce the number of rate switches at a time when the available bandwidth in the network cannot support higher streaming rate. The exponential back-off is performed as follows:

$$T_E^{i+1} = \min\left(\gamma T_E^i, T_E^{\max}\right) \quad (6)$$

where $T_E^{\max}$ is the maximum *Inter-Experiment* timer, and γ is the back-off factor. We clamp the back-off at a maximum to guarantee the sender will periodically probe for spare bandwidth. The *Inter-Experiment* timer of the new stream is rested to initial value T_E^{init}, when the switch experiment to this stream succeeds. The switching experiment duration T_S starts with an initial value T_S^{init} and is updated using an exponential moving average of the time difference between starting a switching experiment to the failure detection time. If no congestion is detected for a duration of T_S seconds, then the server decides to stay at this higher bandwidth. Otherwise the switching experiment is terminated by switching down.

4.3 Determining the Adaptation Parameters

The performance of our adaptation strategies is controlled by a set of different parameters that include the sampling interval, the buffer drain time parameters α and β, the switch up times T_S^{init} and T_E^{init}, and the exponential back-off parameter γ.
We should keep our sampling interval small so that we can effectively track the network bandwidth variations. However, a small sampling interval leads to larger overheads in system complexity and transmitted bandwidth. In order to tradeoff these conflicting goals, we select the sampling interval based on the available network bandwidth; sample at small intervals when the network bandwidth is high (and the variations are likely to be more rapid), and sample at larger intervals when the network bandwidth is low (and the variations are likely to be less frequent). We can do this by sampling every time we transmit a fixed number of bytes. Empirically, we have determined that sampling every time we transmit ~16000 bytes (since we transmit complete packets) provides a good tradeoff for the adaptation. The other adaptation parameters have been tuned empirically to provide a good visual quality, however we can derive analytical bounds on their values based on the statistical properties of the network and video bit-rates. This is a direction of future research.

4.4 SCTP Send-Buffer Scaling

Based on the network congestion feedback, the SCTP sender uses the variable *CWND* to estimate the appropriate congestion window size, which determines the maximum number of unacknowledged packets in flight in the network at any time (Cuetos, 2001). In addition, SCTP uses a fixed size *send-buffer* to store application data before the data is transmitted. This buffer has two functions. First, it handles rate mismatches between the application sending rate and SCTP's transmission rate. Second, it is used to keep copies of the packets in flight so they can be retransmitted when needed. Since the *CWND* determines the number of in-flight packets from the *send-buffer*, setting the *send-buffer* to be smaller than the *CWND* would reduce the throughput of the flow. On the other hand, setting the *send-buffer* much larger than

the *CWND,* will cause more packets to be delayed in the *send-buffer* until they get chance to be sent in the network when acknowledgments have been received for the previous packets in the buffer. This will lead the application to lose control over quality adaptation, as it has to wait longer before making its quality adaptation decisions. In addition, fixed buffer size can introduce significant latency into the SCTP stream, as the packets have to sit in the *send-buffer* until they get chance to be sent in the network (Goel, 2002).

We propose to dynamically adapt the *send-buffer* size to be at least *CWND* packets. However, if the *send-buffer* size is limited to *CWND*, then 1) SCTP must inform the application when it has available space for more packet(s), as well as when it increases the *CWND*, and the application must write the next packet(s) before SCTP can send it. Thus, system timing and scheduling behavior can affect SCTP throughput. 2) Back-to-back acknowledgment arrivals exacerbate this problem. These adverse effects throughput, and can be reduced by adjusting the buffer size so that it is larger than *CWND*. We selected to set the *send-buffer* size to be 2 * *CWND,* as shown in equation (7), which ensures that the SCTP has a window worth of unsent data to keep the self-clock of acknowledgments flowing (Semke, 1998).

$$\text{SCTP send-buffer}(t) = 2 * \text{CWND}(t) \quad (7)$$

The sending application uses non-blocking write calls to ensure that the application is not blocked while there is no space in the SCTP buffer to accept more data, and the data stays in the application buffer. Thus the application buffer size will accurately reflect the current network conditions.

5 PERFORMANCE ANALYSIS

In order to examine our proposed adaptation strategies, we implemented them in Opnet network simulation tool (Opnet). We used the topology shown in Figure 4, where we assume that source S1 is representing the video server, while receiver R1 represents the video client, and S1 is using SCTP for streaming the video to the R1. Sources S2 – Sn are generating traffic that share the bottleneck A-B with the video stream. Unless specified otherwise, we assume that the bottleneck bandwidth is 5 Mbps and the Round Trip time (*RTT*) is 10 ms.

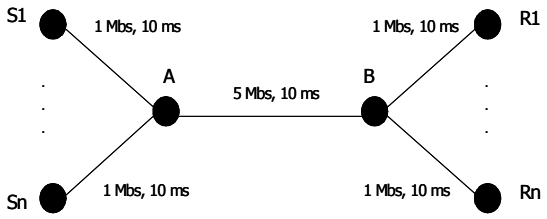

Figure 4: Simulation topology.

For the source S1, we use a XviD video codec (Xvid) to encode a 320×240 - 15 frames per second (fps) video sequence at different bit-rates, V_j (512, 420, 335, 255, 210, 170 Kbps). The adaptation mechanism parameters are: N = 6, D = 3 sec , $\gamma = 2$, $T_E^{init} = 10$ sec, $T_E^{\max} = 60$ sec, $T_S^{init} = 10$ sec. Additionally, we select the parameters $\alpha = 0.4$, and $\beta = 0.5$. We measure the video quality in terms of the achieved average bit-rate, variations in the quality at the client, and data loss due to prefetch buffer starvation which can result in frame drops or pauses in the video to allow for rebuffering. Hence, we want to track the available bandwidth faithfully, while minimizing prefetch buffer underflow and maintaining a relatively constant quality by minimizing the number of stream switches.

5.1 Bandwidth Adaptation

We examined the adaptation mechanisms by real network trace, obtained from the PlanetLab (Planetlab), between two nodes one in the US east coast and the other in the west coast. The traces were collected over 70 minutes with 50 competing TCP connections and the bandwidth varied between 700 Kbps and 120 Kbps. We show the adaptation performance over 3000 seconds in Figure 5. As expected the instantaneous decision leads to over-aggressive switching down, thereby not following the network trace accurately, unlike the combined decision strategy. However, as we have mentioned before, bandwidth fidelity is not the only performance metric we evaluate. We also measure the number of stream switches and the number of dropped packets (due to server buffer overflow).

5.2 Effect of SCTP Send-Buffer Scaling

To examine the combined adaptation mechanism with the SCTP *send-buffer* scaling, we compared the adaptation mechanism, with and without the *send-buffer* scaling. To vary the bandwidth between

sender S1 and receiver R1, we varied the number of active SCTP connections, as shown in Table 1.
Figure 6 shows the bandwidth available to the SCTP connection between S1-R1, as well as the video stream rate received at the client R1. The graph shows that although the adaptation mechanism without the buffer scaling is able to track the available bandwidth, using the buffer scaling will We assumed that the SCTP connections will always have data to send. For each set of SCTP streams we run the experiment with and without the SCTP *send-buffer* scaling. We calculated the normalized SCTP allow the application to track the available bandwidth more accurately, as the server buffer will be more reflective to the SCTP available bandwidth than the using a fixed size for the SCTP *send-buffer*. To ensure that the SCTP *send-buffer* scaling will not affect the SCTP throughput We run different number of SCTP streams through the bottleneck A-B, and we set all the connections to last for 200 seconds. throughput as the average throughput of a SCTP connection with the buffer scaling option to that without the buffer scaling.

Figure 7 shows the normalized SCTP throughout, with 90% confidence interval, versus the number of active SCTP connections. The figure shows that the SCTP buffer scaling will not have an adverse effect on the SCTP throughput.

Table 1: Number of active SCTP connections.

Simulation Time (Sec.)	Number of SCTP Connections
0	25
15	20
30	40
70	20

In Figure 8, we examined the video stream goodput as a function of the *RTT*. We define the goodput as the percent of video packets that arrive before the display time of their video frame at the client to the total number of video packets sent from the server.

In all the experiments we set the client pre-buffering to 3 seconds. Results show that without quality adaptation 24% to 62% of the non-adaptive stream packets will miss their deadlines at the client. The reason behind this that the video sender is not adapting its rate to the available bandwidth, which leads many packets to be delayed at the sender and to miss their deadlines. This is reflected at the client as buffer underflow events, that leads the displayed video to continually freezes.

6 CONCLUSIONS

We proposed server adaptation mechanism, for low-delay video over AIMD transport protocols. The server estimates information about the available network bandwidth by monitoring the application buffer and performs stream switching to meet bandwidth and delay constraints. We investigate a strategy that combined instantaneous and look-ahead strategies for switching down the transmission rate, and switch up the rate in a controlled manner after observing periods of no congestion. We estimate the current and future server buffer drain delay, and derive the transmission rate to minimize client buffer starvation. In addition, we presented a simple scaling mechanism to the transport protocol *send-buffer* that allows the adaptation mechanism to follow accurately the network bandwidth and reduce the adaptation decision time. We implemented these algorithms over SCTP. The strategy with combined look-ahead and instantaneous decisions can follow the network bandwidth accurately. In addition, the proposed SCTP *send-buffer* scaling does not affect the SCTP throughput, compared to SCTP with a fixed buffer size, while it improves the accuracy of

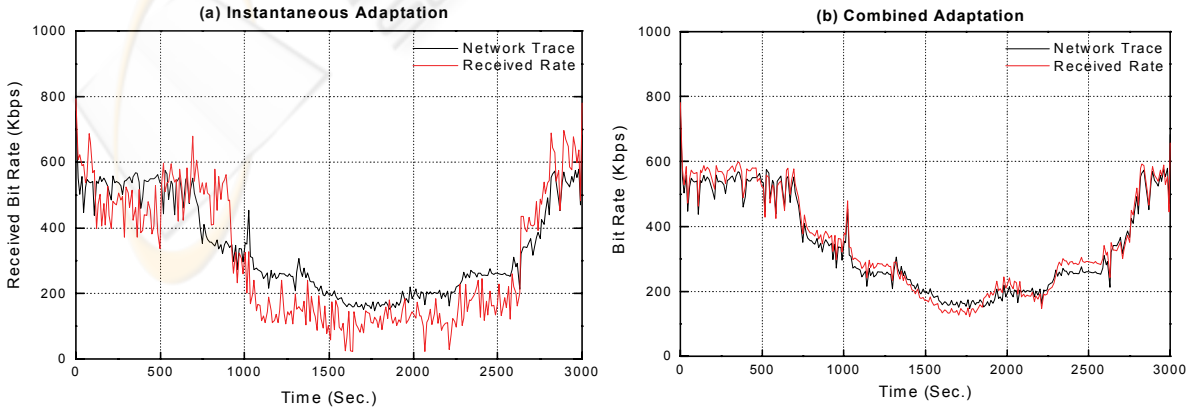

Figure 6: Received rate vs. available bandwidth.

the quality adaptation.

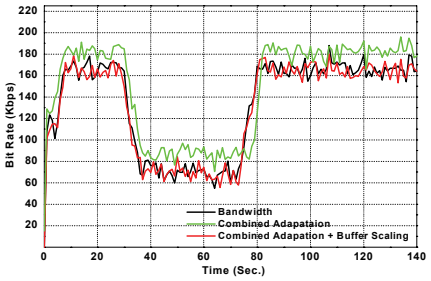

Figure 5: Adaptation performance for real trace. trace.

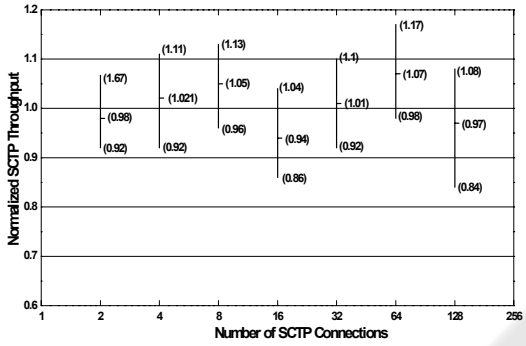

Figure 7: Normalized SCTP throughput versus number of connections.

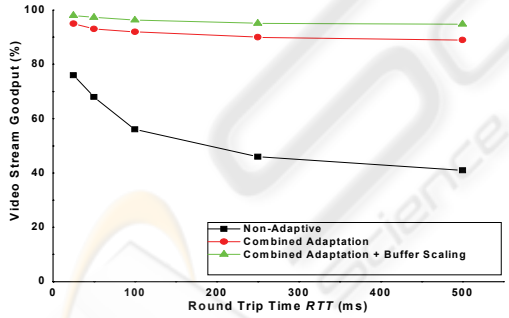

Figure 8: Video stream goodput vs. round trip time.

Disclaimer

The views and conclusions in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research laboratory or the U.S. Government.

REFERENCES

Krasic, C., Li, K., Wapole, J., 2001. The case of streaming Multimedia with TCP. In *proc, of the 8th Int. Workshop on Interactive Distributed Multimedia Systems (iDMS)*.

Hsiao, P., Kung, H., Tan, K., 2001. Streaming Video over TCP Receiver-based Delay Control. In p*roc. of ACM NOSSDAV*.

Cuetos, P., Saparilla, D., Ross, K., 2001. Adaptive Streaming of Stored Video in a TCP –Friendly Context: Multiple Versions or Multiple Layers ?. In *Int. Packet Video Workshop*.

Stewart, R., Xie, Q., et al., 2000. Stream Control Transmission Protocol, RFC 2960.

Blak, A., Sigler, M., Gerla, M., Sandidi, M., 2002. Investigation of MPEG-4 Video Streaming over SCTP. *In proc. of SCI*.

Brennan, R., Curran, T., 2001. SCTP Congestion Control: Initial Simulation Studies. In *proc. of Int. Teletraffic Congress CFP*.

Mehra, P., Zakhor, A., 2003. TCP-Based Video Streaming Using Receiver-Driven Bandwidth Sharing. In *proc. of the 13th Int. Packet Video Workshop*.

Kanakia, H., Mishra, P., Reibman, A., 1993. An adaptive congestion control scheme for real-time packet video transport. *SIGCOMM Symposium on Communications Architectures and Protocols*.

Saparilla, D., Ross, K., 2000. Streaming Stored Continuous Media over Fair-Share Bandwidth. *Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*.

Goel, A., Krasic, C., Li, K., Walpole J., 2002. Supporting Low Latency TCP-Based Media Streams. In *proc. of the Tenth Int. Workshop on Quality of Service.*

,

Mahdavi, J., Mathis, M., 1998. Automatic TCP Buffer Tuning. *Computer Commu nication Review, ACM SIGCOMM*, volume 28, number 4, Oct. 1998.

Semke, J., Mahdavi, J., Mathis, M., 1998. Automatic TCP Buffer Tuning. *Computer Commu nication Review, ACM SIGCOMM*, volume 28, number 4, Oct. 1998.

Opnet Network Simulator, www.opnet.com

XviD video codec, www.xvid.org

PlanetLab wide area network testbed, http://www.planet-lab.org/