Journal of Internet Technology 5(4), pp. 331-339, 2004

A Two-Stream approach for priority management and adaptive rate control in multimedia applications

Longin Jan Latecki
Dept. of Computer and Information Sciences
Temple University
Philadelphia, PA 19094

latecki@temple.edu

Tao Jin
Dept. of Computer and Information Sciences
Temple University
Philadelphia, PA 19094

thomasjt@temple.edu

Jaiwant Mulik
Dept. of Computer and Information Sciences
Temple University
Philadelphia, PA 19094

jmulik@temple.edu

ABSTRACT

In this paper, we propose a two-stream approach for adaptive rate control in multimedia applications. By monitoring a low-rate *monitoring stream*, we keep track of the available bandwidth (AB) of the network path and dynamically adjust the sending rate of the *traffic stream* close to the optimal rate. We also assure the higher priority of the monitoring stream in that the traffic stream is not allowed to affect the transmission of the monitoring stream. The proposed two-stream approach perfectly meets the requirements of the current best-effort Internet and fits well in multimedia applications. For example, there is no bandwidth overhead for the monitoring stream in peer-to-peer video conferencing, because the monitoring stream is the audio stream. We show in our experiments that both the network and the application can benefit from this approach. The proposed two-stream approach is applicable to monitor the sending rate of the traffic stream over UDP as well as over TCP.

Keywords: Adaptive rate control, TCP friendly congestion control, audio and video streaming, video conferencing, video telephone.

1 Introduction

We present a system for continuous adaptive rate control that is useful for real time multimedia streaming, particularly video telephone over the public Internet. No changes to existing transport protocols are required, since it is possible to incorporate our system at the transport layer.

We ensure that audio stream has absolutely higher priority, and transmit video stream using only the remaining bandwidth. Steinmetz [9] has shown that during a multimedia session over a congested network it is, in terms of human perception, more important to maintain a continuous (minimum jitter) audio stream than a video stream. Our system makes real time transmission of video over TCP possible since the one-way delay when video stream is transmitted over TCP with our rate control is on the same level as for video over UDP. We performed a large number of experiments both in the controlled environment as well as on real networks that included a dialup 28.8 kbps and cable modem connections. These experiments verify excellent performance of the proposed system.

Our main contributions are several substantial extensions and modifications of Pathload [1]. They were necessary since Pathload is designed for one-time bandwidth estimation, while we need continuous bandwidth monitoring in the interaction with the sender rate control. Following Pathload, the available bandwidth (AB) is detected at the application level by making use of the *One Way Delay (OWD) property under congestion*. On the level of OWD analysis, we propose a multiscale analysis of OWD properties and a method for detection of decreasing trend (Pathload uses only increasing trend). On the rate control level, we extend the Pathload iterative rate control algorithm to allow for instantaneous and immediate bandwidth adaptation for the traffic stream.

Our modeling framework is different from Pathload. We use the two-stream model introduced in Latecki et al. [3]. All network measurements are performed on so-called *monitoring stream*, called stream A for audio, which is assumed to be substantially smaller than the AB, while the main data transport stream is called *traffic stream,* called stream V for video. The measurements preformed on monitoring stream are used to adjust the sending rate of the traffic stream.

2 Related Work

The area of streaming multimedia has been extensively researched for several years. In this section, we compare our approach with a representative selection of earlier approaches.

Jain and Dovrolis [1][2] propose a Self-Loading Periodic Streams (SLoPS) approach and a tool named Pathload to detect the AB of the network path based on the OWD property under congestion. Pathload sends periodic streams into the network path and detect the OWD trends at the receiver side. The congestion detection in Pathload

is based on detection of increasing trend in the OWDs. Two tests called PCT and PDT are used for OWD increasing trend detection (Section 3.1).

Bansal and Balakrishnan [6] present a family of innovative TCP congestion control algorithms called binomial algorithms. These algorithms prevent a drastic reduction in transmission rate upon congestion and are designed for streaming audio and video applications. They show that there exist infinitely many deployable TCP-friendly binomial algorithms. Our approach differs from [6] in two important aspects. First, we consider a multimedia application as a whole consisting of simultaneous audio and video streams with an explicit hierarchy between them. In the event of congestion, our goal is to maintain the transmission rate of the audio stream while sacrificing the video stream. Second, we try to prevent the packet loss as an indicator of congestion in TCP, by using delay trends in the audio stream as our congestion signaling mechanism. Third, they require changes to the transport layer, while we do not.

Cen et al. [7] describe the Streaming Control Protocol (SCP) which is a TCP-like and TCP-friendly transport protocol designed to prevent the abrupt rate changes of TCP. However, the SCP does not allow inter-stream state sharing that our approach uses in order to provide a priority to the audio stream over the video stream.

Following the results of Steinmetz in [9], Ghinea at al. [8] present the Dynamically Reconfigurable Protocol Stacks (DRoPS) architecture to dynamically reconfigure protocol elements in order to achieve maximum Quality of Perception (QoP). QoP is a subjective user side measure of perception. They also developed an analytic model relating traditional QoS measures to a QoP measure. The parameters of this model are application dependent. While DRoPS is an elaborate architecture with a Linux Kernel implementation our approach is lightweight and implemented entirely in user space. Also, [8] reports that in applications with more perceptual weight to audio, such as the videoconferencing application considered in this paper, the TCP/IP stack (the one we used) performs better than DRoPS.

3 Two-Stream Approach

Basic idea

The basic idea is to use a low rate life-time *monitoring stream* to keep track of the ever-changing network, in order to find the optimal sending rate (close to the end-to-end AB) for the *traffic stream*. The *monitoring stream* must be sent over UDP because the mechanism is based on the OWD properties of the stream packets, and only for UDP packets can we measure the packets OWD within the application layer. The *traffic stream* can be sent over any protocol.

We continuously detect OWD trends in the *monitoring stream*, use them as an indication of the relationship between the current traffic rate and the AB, and adjust the transmission rate of the *traffic stream* close to the AB. By assigning the low-rate audio stream as the *monitoring stream*, we can make use of the information naturally contained in the in-band traffic without introducing any intrusive traffic. In our approach video can be sent over either UDP or TCP, since in both cases, the monitoring stream keeps video rate below the AB, preventing losses (UDP and TCP) and subsequent retransmissions (TCP). We assume that there exists sufficient bandwidth to transmit the *monitoring stream*.

Improved performance when video is sent over UDP

Since UDP is unresponsive to AB, in cases of constrained bandwidth, not only the video stream suffer due to packet loss but also the quality of audio will drop significantly due to the congestion caused by the video stream [3].

By using our adaptive rate control mechanism, it is possible for the application to continually estimate the AB and set the optimal rate for the video stream, significantly improving the quality of both audio and video.

Improved performance when video is sent over TCP

If the video is sent over TCP, the TCP congestion control mechanism will constrain the actual transmission rate on the network, preventing the network from getting congested. However, TCP's reliable transmission will cause delay to accumulate, thus make it unsuitable for real time multimedia applications. One way to alleviate the accumulated delay is to drop the obsolete frames at the sending side. But most of the delay still could be very serious if we do not choose an appropriate video rate (Figure 3). Our adaptive rate control solves this problem by adjusting the video rate slightly below the AB, which makes TCP retransmission rate close to zero. This way we make real time video transmission over TCP a good option. The advantage of reliable video transport is a significant improvement of the received video quality. A study showing advantages of partially reliable video transmission is presented in [10].

3.1 Detecting OWD Trends in the Continuous Monitoring Stream

The OWD (one way delay) of a packet stream will increase when the traffic rate is above the end-to-end AB [2].

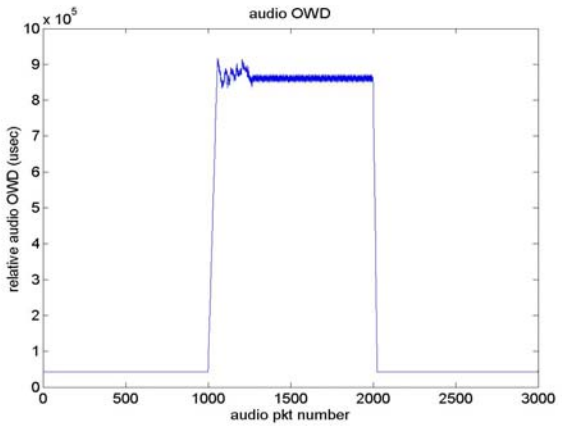

Figure 1: OWD property under congestion

In an experiment illustrated in Figure 1, the end-to-end AB before we sent any traffic was set to 100 kbps. We sent the audio stream at a constant rate of 20.8 kbps, and the video stream over UDP at a rate of 120 kbps from audio packet number 1000 to 2000. The OWD was measured in the audio stream. As can be seen in Figure 1, that is a short transition phase in which the OWD shows obvious increasing trend at the beginning when the network become congested. If we detect this transition phase, we can respond to the congestion even before packet loss happens. Our two-stream approach for adaptive rate control is based on this fact.

OWD phases

OWDs of the stream packets show different patterns under different network conditions. We can divide the whole process into the following four types of phases based on the congestion status and OWD trends. R is transmission rate of traffic stream

- *Increasing phase* (R > AB): The short transition period before entering the congestion. The OWDs show increasing trend.
- *Decreasing phase* (R < AB): The short transition period when recovering from congestion. The OWDs show decreasing trend.
- *Steady phase* (R < AB): The OWDs are stable in this phase.
- *Congested phase* (R > AB): The network is already congested. Even though usually more variant than in the steady phase, the OWDs in this phase are stable.

OWDs are stable in both the *steady* and *congested phases*, so it is hard to discriminate these phases based on the measured OWDs, especially when we only have a small window of measurements in real time environment. Thus, it is crucial that we detect the *increasing phase* before the traffic goes into the *congested phase*.

PCT and PDT

Two complementary statistic metrics named PCT and PDT can be used to detect the OWD trends in the stream. Before calculating the PCT and PDT from K measured OWDs, we pre-process the data because they usually contain a lot of noise and outliers. The K OWDs $\{D_1, D_{2,...,} D_K\}$ are partitioned into Γ groups, each group use the median value $\hat{D}_i$ as its representative value. K is also known as the *detection period*.

The original *Pairwise Comparison Test* (PCT) metric of a stream was defined in [2] as

$$S_{PCT} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}_k > \hat{D}_{k-1})}{\Gamma - 1},$$

where I(X) is one if X holds, and zero otherwise.

This metric is only sensitive to the OWD increasing trend in the stream. But as described in Section 3.2, we also need to detect the decreasing phase in the stream. So we introduce and extended PCT as

$$S_{PCT} = \frac{\sum_{k=2}^{\Gamma} I(\hat{D}_k, \hat{D}_{k-1})}{\Gamma - 1},$$

where,

$$I(\hat{D}_k, \hat{D}_{k-1}) = \begin{cases} 1, & if\ \hat{D}_k - \hat{D}_{k-1} > \varepsilon \\ 0, & if\ \left|\hat{D}_k - \hat{D}_{k-1}\right| \le \varepsilon \\ -1, & if\ \hat{D}_k - \hat{D}_{k-1} < -\varepsilon \end{cases}$$

ε is a predefined threshold value related to the granularity of the increasing or decreasing step.

If the OWDs are independent, the expected value of S_{PCT} is 0. If there is a strong increasing trend, S_{PCT} approaches 1. If there is a strong decreasing trend, S_{PCT} approaches -1. Due to our modification, the modified S_{PCT} can detect both increasing and decreasing trends. Moreover, by defining the threshold ε, it is more stable and noise resistant.

The *Pairwise Difference Test* (PDT) metric of a stream is defined in [2] as

$$S_{PDT} = \frac{\hat{D}_{\Gamma} - \hat{D}_1}{\sum_{k=2}^{\Gamma} |\hat{D}_k - \hat{D}_{k-1}|}$$

If the OWDs are independent, the expected value of S_{PDT} is 0. If there is a strong increasing trend, S_{PDT} approaches 1. If there is a strong decreasing trend, S_{PDT} approaches -1.

S_{PCT} and S_{PDT} are complement. S_{PCT} detects the ratio of the increasing and decreasing steps, while S_{PDT} quantifies how

strong the start-to-end variation is. S_{PCT} and S_{PDT} are combined to determine the current OWD phase. "Increasing phase" and "decreasing phase" are reported when either of the two metrics detects it, and the other one does not disagree, while "steady phase" is reported only when both of the two metrics indicate a "steady phase". All the other situations are reported as "ambiguous phase".

We define two thresholds L_{PCT} and U_{PCT} for S_{PCT}, and another two L_{PDT} and U_{PDT} for S_{PDT}. Our decision rules are summarized the following:

- If $S_{PCT} > U_{PCT}$ and $S_{PDT} > L_{PDT}$, or $S_{PDT} > U_{PDT}$ and $S_{PCT} > L_{PCT}$, "increasing phase" is reported
- If $S_{PCT} < -U_{PCT}$ and $S_{PDT} < -L_{PDT}$, or $S_{PDT} < -U_{PDT}$ and $S_{PCT} < -L_{PCT}$, "decreasing phase" is reported
- If $-L_{PCT} < S_{PCT} < L_{PCT}$, and $-L_{PDT} < S_{PDT} < L_{PDT}$, "steady phase" is reported
- "ambiguous phase" is reported in all other situations.

The thresholds can be adjusted depending on how sensitive PCT and PDT should be. In our experiments, L_{PCT} and L_{PDT} were set to 0.25; U_{PCT} and U_{PDT} were set to 0.5.

With these two statistic metrics, we can detect the current OWD phase in the monitoring stream, and use it as an indicator of the relationship between the transmission rate and AB. Figure 2 illustrates the detected PCT (line with 'x' marks) and PDT for the OWDs in Figure 1. PCT and PDT were calculated every 32 measured OWDs in the audio stream. One obvious observation from Figure 1 andFigure 2 is that the OWD transition phase is usually very short. When the traffic rate is above the AB, the OWD will quickly increase until congestion sets in. After this short transition phase, the OWD will remain relatively stable, hence using PCT and PDT it is hard to distinguish whether or not the network is congested. However since we can detect the *increasing phase*, the *congested phase* will never appear without a correct rate adjustment, so it is tolerable that we cannot detect the *congested phase*

.

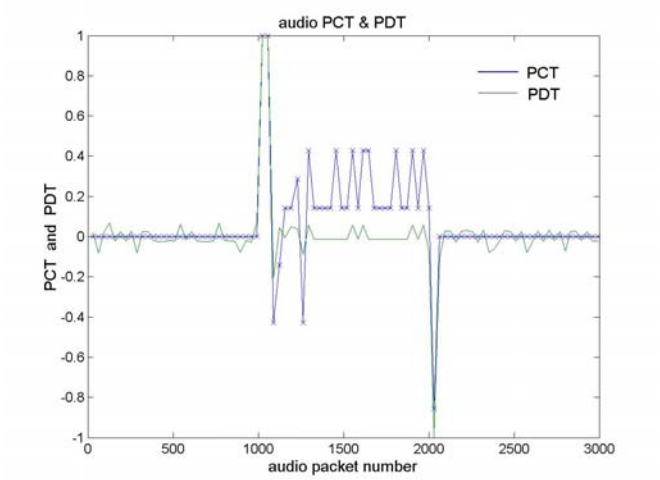

Figure 2: Graphs of PCT PDT for OWD in Figure 1

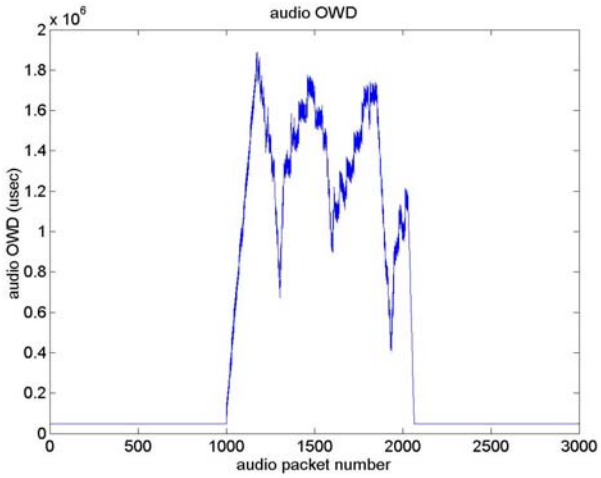

Figure 3: Audio OWD when video sent over TCP

The *detection period*

A suitable length of *detection period* K is critical in detecting the OWD trends in the continuous monitoring stream. In order for the PCT and PDT metrics to be able to detect the OWD transition phases (increasing and decreasing phases), it is important that the detection period is within the OWD transition phase. In other words, if the detection period only covers part of the transition phase, it may not able to detect it. In this sense, the detection period should be as short as possible; but on the other hand, the detection period cannot be too short, otherwise the increasing trend will be too minor to be detected.

Experiments show that the length of the OWD transition phase vary greatly under different conditions, e.g., when the traffic rate is far above the end-to-end AB, the OWD quickly increases to the peak. When the traffic rate is

slightly above the AB, the increasing phase will be longer. Experiments also show that when the traffic stream is sent over TCP, the OWDs usually only show long-term increasing (or decreasing) trends and display great local variance (Figure 3).

Our experiments indicate that a short detection period works well under UDP environment, while a longer detection period is more suitable when the traffic is sent over TCP.

The setting of the experiment corresponding to Figure 3 was the same as of the experiment in Figure 1, except that the video was sent over TCP. The sender tried to send the video stream at rate 100 kbps, but was regulated by the TCP congestion control mechanism. The OWDs in the audio stream clearly shows the TCP's slow-start and AIMD (Additive Increase Multiplicative Decrease) effects. Compared with Figure 1, the OWD transition phases are much longer, and the OWDs are more variant in small scale during the transition phases.

Multiscale and Sliding-window Measurement

Since it seems impossible to decide an appropriate *detection period* that can work under all circumstances, we use a multiscale and sliding-window method to detect the OWD phases in the *monitoring stream*.

Multiscale means that we calculate the PCT and PDT over multiple scales of detection periods simultaneously. Figure 4 illustrates 3-scale measurements over the OWDs displayed in Figure 3. In scale 1, PCT and PDT were calculated every 32 packets; in scale 2, they were calculated every 64 packets; and every 128 packets in scale 3. Scale 1, in most cases, would not report the increasing or decreasing phases, because either the PCT or PDT could not satisfy the thresholds (see the decision rules in Section 3.1). The fundamental reason is that the detection period was so small so that the measurements were dominated by the noises. Scale 2 seems better, but still would miss a lot if the thresholds were set a little bit high. Scale 3 very nicely caught all the increasing and decreasing trends and would report them correctly.

The final decision is based on the 3-scale measurements. The decision rules are summarized as follows

- If any of the 3-scale measurements reports "increasing phase", and the other two do not disagree (i.e., report "decreasing phase"), then "increasing phase" is reported
- If any of the 3-scale measurements reports "decreasing phase", and the other two do not disagree (i.e., report "increasing phase"), then "decreasing phase" is reported
- If at least 2 of the 3-scale measurements report "steady phase", then "steady phase" is reported
- Otherwise "ambiguous phase" is reported

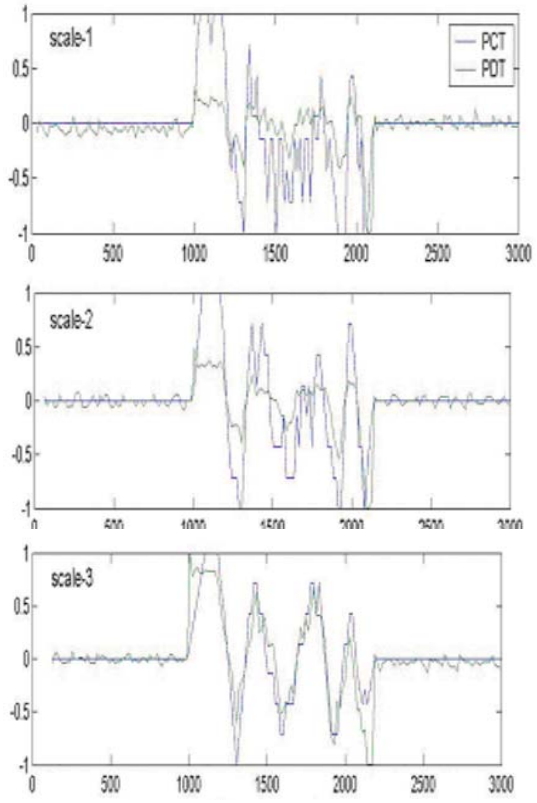

Figure 4: 3-scale PCT and PDT detection

Figure 5 shows the final result from the 3-scale detection in Figure 4. Here we use 1 to represent "increasing phase", -1 to represent "decreasing phase", 0 to represent "steady phase", and 0.5 and -0.5 to represent "ambiguous phase" (-0.5 means "more probably a decreasing phase", and 0.5 means "more probably a increasing phase"). We can see that Figure 5 perfectly matches the OWD phases corresponding to Figure 3.

We also use sliding and overlapping windows to make the measurements more frequent without making the detection period too short.

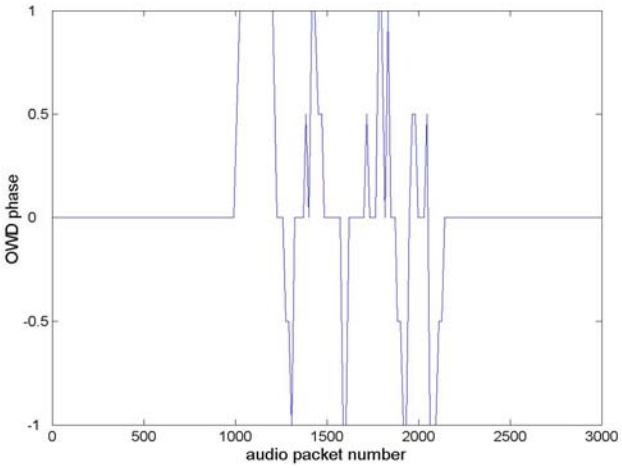

Figure 5: Final result from 3-scale detection in Fig. 4

3.2 An Interactive Adaptive Rate Control Algorithm in the Two-Stream Environment

The advantage of having a lifetime *monitoring stream* is that we can detect the AB at any moment in the traffic.

The adaptive rate control mechanism is designed to be able to detect congestion and adjust the sending rate close to AB at any moment of the traffic lifetime. The whole process of the traffic regulation consists of two alternative stages: *rate-adaptation stage* followed by *steady stage.* The algorithm searches for the AB during the *rate-adaptation stage*. Once the sending rate converges to the optimal rate, it enters the *steady stage*. During the *steady stage*, we still continuously keep track of the network status through the *monitoring stream*. If there is any congestion detected during the *steady stage*, or we decide to try a higher rate, the traffic changes into the *rate-adaptation stage* and search for the AB again.

The algorithm used for searching the AB in the *rate-adaptation stage* is very similar to the iterative algorithm used in the Pathload [1]. We denote the sending rate at time *n* as *R(n)*, the lower and upper bounds for the AB as R_{min} and R_{max}. At time *n,* we calculate R(n+1) as:

If $R(n) > AB$, $R_{max} = R(n)$;

If $R(n) \leq AB$, $R_{min} = R(n)$;

$R(n+1) = (R_{max}+R_{min})/2$;

But we need to do the following adjustments, because our traffic is continuous while Pathload only sends short streams.

- When an "increasing phase" is reported (R(n) > A), we add a *break* in traffic stream before trying the next sending rate, so that the OWD can drop to the normal level (the queues in the routers can be cleaned) before we send more traffic
- The *break* last as long as the "decreasing phase" is detected. We resume sending of the traffic stream when the decreasing phase is finished
- When "ambiguous phase" is reported for sufficiently long time, we regard it as a symptom of slight congestion, so slightly decrease the sending rate

The *break period* is needed so that two consecutive increasing tends can be detected. Also, we wait until the *decreasing phase* disappears since it means that the network is still recovering from congestion.

4 Experimental Results

We present some experimental results that illustrate the benefits of our adaptive rate control for both the network and the multimedia application. We performed a large number of experiments both in the controlled environment as well as on the real networks that include a dialup 28.8 kbps and cable modem connections. We present here our experiments conducted in Emulab (Netbed) [5] environment, since the obtained results are representative for all our experiments (including experiments on real networks) and the experiments on Emulab can be verified independently. A detailed report from our experiments (including experiments on real networks) and the program source code can be obtained from http://www.cis.temple.edu/~latecki/ARC.

4.1 Experimental Setting

The bandwidth of the tight link between the sender host SEND and receiver host RECV was set to 100 kbps. When there is no cross traffic over the tight link, the end-to-end AB of the path is 100 kbps. We simulated the two-stream traffic from SEND to RECV, in uni-direction. The audio (monitoring) stream was sent over UDP, at a constant transmission rate of 20.8 kbps, and the packet interval was 40 ms. The video frames were generated every 20 ms, and the SEND sent the frames as soon as they were generated. The video frame size can be controlled by the application. For video sent over UDP, the actual transmission rate on the network can be easily calculated from the frame size, because each frame will result in a UDP packet on the network. So the transmission rate is

(video frame size + UDP header size + IP header size + Ethernet header size) / 20 ms.

All the experiments followed the same scenario:

1. We first streamed just audio data for 40 seconds (1000 audio packets)
2. Then we sent the video stream for 120 seconds (between audio packet number 1000 and 4000)

3. Then we stop video stream and sent only audio packets for another 40 seconds

We used both UDP and TCP to transmit video stream. For each of the protocols, we performed two runs of the experiments, one without rate control and the other with our adaptive rate control, and compared the results. After applying our adaptive rate control, both streams' loss rates dropped to zero after the optimal rate was found. When cross traffic was added, no matter sent over TCP or UDP, all the experiments showed similar results. In Section 4.2 we present our experiments when video was sent over TCP. The results for UDP are omitted, since streaming video over TCP seems to be a more interesting case.

4.2 Video Stream Sent Over TCP

Figure 6 shows the audio and video stream rate from the application when adaptive rate control was not applied. Figure 7 shows the actual video transmission rate on the network (due to the TCP congestion control). For easy comparison, we plot the video rate with respect to audio packet number. Figure 8 displays the two streams' rates from the application when the adaptive rate control was applied. We see from Figure 8 that it took about 20 seconds for the video stream rate to converge to 48.8 kbps (actual transmission rate was a little higher than the stream rate). Figure 9 shows the actual video transmission rate on the network sent over TCP when the adaptive rate control was applied on the application level. Observe by comparing Figure 8 and Figure 9 that the actual video sending rate from the application matches the TCP sending rate when our rate control runs on the application layer. This is not the case if there is no rate control on the application layer (Figures 6 and 7).

Figure 10 compares the audio OWDs in the two experiments. While the mean of the audio OWDs during video transmission without rate control increased by 1221 ms (from 53 ms to 1274 ms), it increased only by 2 ms (from 55 ms to 57 ms) with our rate control (in comparison to the first phase when only audio packets were sent). We can clearly see that the TCP increasing trend is reflected in OWD of audio packets by comparison of the audio OWD without rate control in Figure 10 to video rate sent over TCP in Figure 6.

Figure 11 compares the video OWDs (we measured the frame delay in the application layer) in the two experiments. While the mean of the video OWDs was 8409 ms without rate control, it was only 70 ms with our rate control (most in the rate-adaptation stage), with the difference between minimum OWD and mean OWD of 11 ms. Both audio and video delays are acceptable for real time audio-video communication [4]. Thus, the proposed rate control makes it possible to have video transmission over TCP in real time, peer-to-peer video conferencing.

Figure 12 shows the loss rate change in the audio stream. While 143 out of the total 3000 audio packets were lost during the video transmission without rate control (average loss rate was 4.77%), 0 packet was lost after our rate control was applied in the video stream. Figure 13 shows the drop rate of the video stream (SEND makes the best effort to send frames, and drops frames when they become obsolete). Without rate control, there were 1064 out of the total 6000 video frames were dropped (average drop rate was 17.75%). When the rate control was applied, 302 frames were dropped only in the rate-adaptation stage. The overall drop rate in the video becomes 0 when the traffic is sufficiently long.

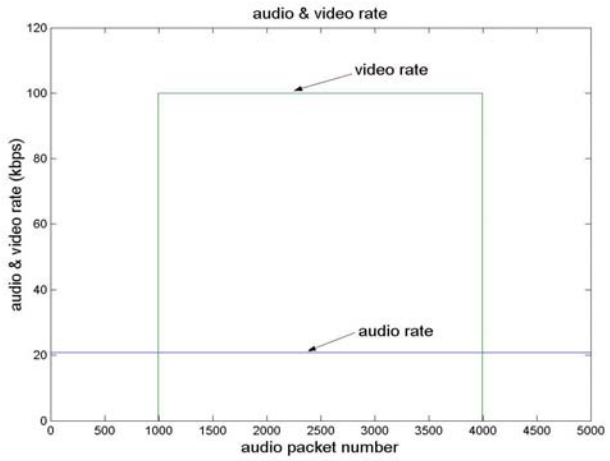

Figure 6: Audio and video rate (without rate control)

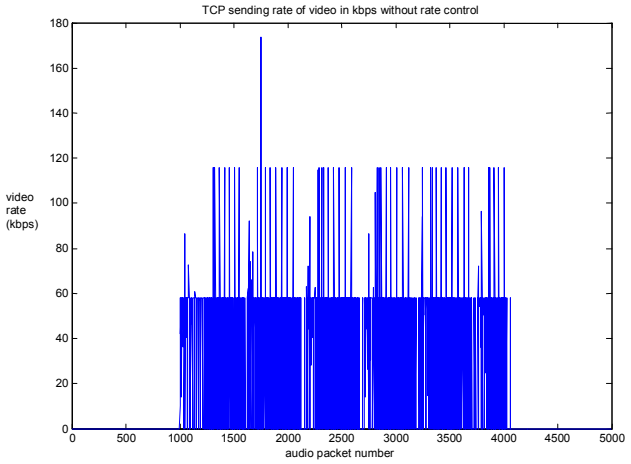

Figure 7: Video rate in kbps sent over TCP without rate control

5 Conclusions

The proposed adaptive rate control algorithm is suitable for real time multimedia streaming over UDP as well as over TCP. In particular, it significantly improves real time performance of TCP congestion control, so that real time video transmission over TCP is possible. The reason is that our rate control (that does not introduce any packet loss) suppresses TCP rate control, and consequently the packet loss due to the TCP increasing trend is kept very close to zero. Our experiments demonstrate that not only the performance of the video stream but also the audio stream is drastically improved. Our real network experiments indicate that a small amount of retransmissions due to sporadic packet loss (it was below 0.01%) has only minor influence on OWD. We implemented our algorithm on the application layer, but it is also possible to use it on the transport layer, in particular in conjunction with future transport protocols that are more suitable for the multimedia data.

The effect of datagrams within a stream taking different routes Given the datagram forwarding nature of most of the Internet, we should mention that packets not following the same route is a limitation present in most AB measurement tools also. However, we assume that due to several measurements the effects of packet taking different routes in minimized, which is verified by our experiments on the public Internet, and that work on this is part of future work.

6 Acknowledgements

We would like to thank Phillip Conrad for his helpful comments. We are grateful to Jay Lepreau and the support staff of Netbed (formerly known as Emulab), the Utah Network Emulation Testbed (which is primarily supported by NSF grant ANI-00-82493 and Cisco Systems) for making their facilities available for our experiments.

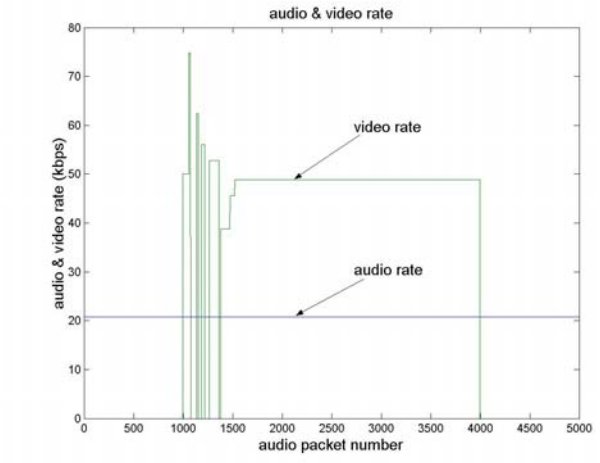

Figure 8: Audio and video rate (with rate control)

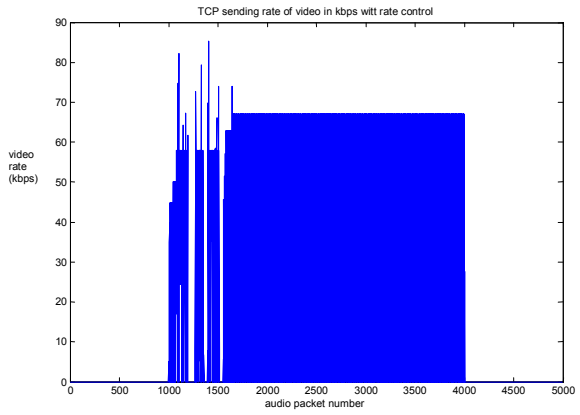

Figure 9: Video rate (kbps) over TCP with rate control

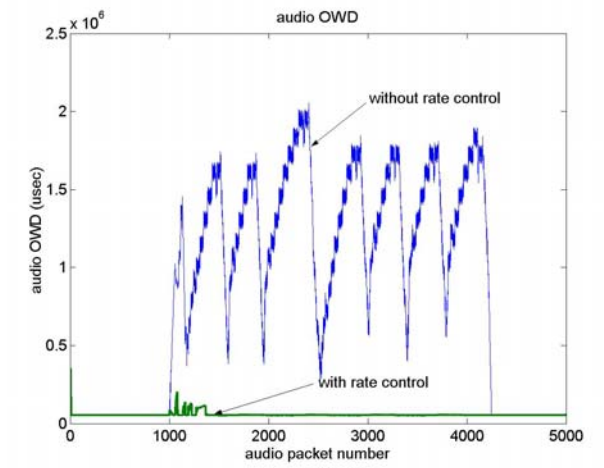

Figure 10: Audio OWD

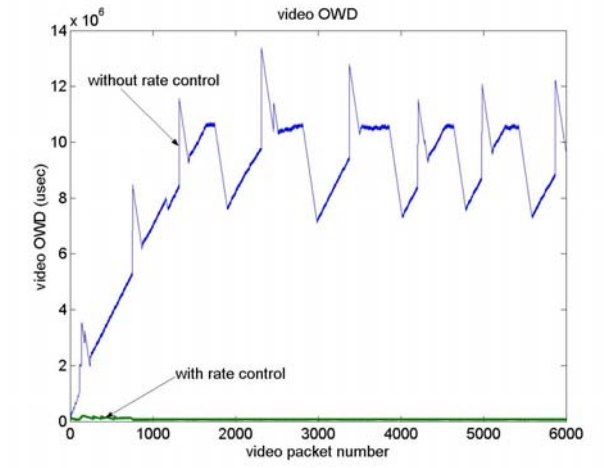

Figure 11: Video OWD

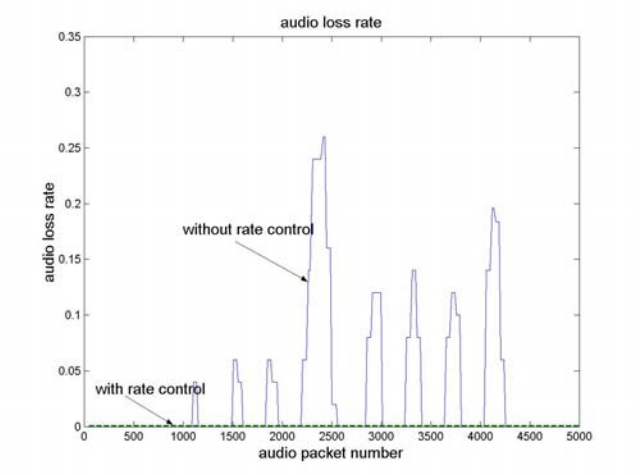

Figure 12: Audio loss rate

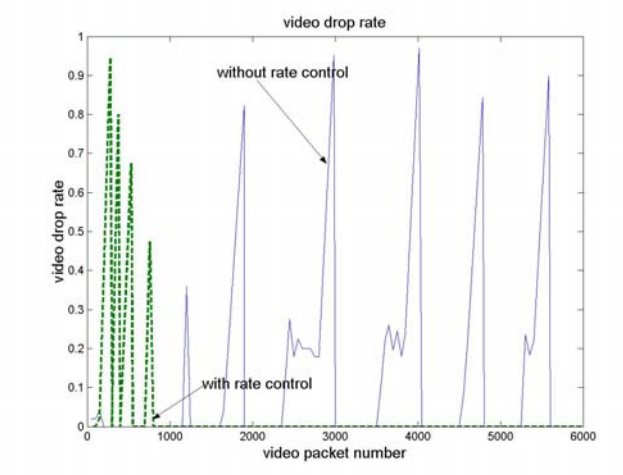

Figure 13: Video drop rate

7 References

[1] Jain, M. and Dovrolis C., "Pathload: A measurement tool for end-to-end available bandwidth." *Proc. of Passive and Active Measurements (PAM) Workshop,* Mar. 2002.

[2] Jain, M. and Dovrolis C., "End-to-End Available Bandwidth: Measurement Methodology, Dynamics, and Relation with TCP Throughput." *Proc. ACM SIGCOMM*, 2002.

[3] Latecki, L. J., Kulkarni, K., and Mulik, J., "Better Audio Performance When Video Stream Is Monitored By TCP Congestion Control." *Proc. IEEE Multimedia and Expo*, July 2003.

[4] Steinmetz, R., "Synchronization Properties in Multimedia Systems." *IEEE J. on Selected Areas in Communications* 8(3), 1990, pp. 401-412.

[5] White, B., Lepreau, J., Stoller, L., Ricci, R., Guruprasad, S., Newbold, M., Hibler, M., Barb, C., and Joglekar, A., "An Integrated Experimental Environment for Distributed Systems and Networks." *Proc. 5th Symposium on Operating Systems Design and Implementation*, 2002.

[6] Bansal, D. and Balakrishnan, H., "Binomial Congestion Control Algorithms." *Proc. IEEE INFOCOM 2001*, Vol. 2, Anchorage, Alaska, 2001, pp. 631-640.

[7] Cen, S., Pu, C., and Walpole, J., "Flow and congestion control for Internet streaming applications." *Proc. Multimedia Computing and Networking (MMCN)*, 1998.

[8] Ghinea, G., Thomas, J. P., R. and Fish, S., "Multimedia, network protocols and users - Bridging the gap." *Proc. ACM Int. Conf. on Multimedia*, Orlando, Florida, 1999, pp. 473 – 476.

[9] Steinmetz, R., "Human Perception of Jitter and Media Synchronization." *IEEE Journal on Selected Areas in Communications* 14(1), 1996, pp. 61 – 72.

[10] Molteni, M. and Villari, M., "Using SCTP with Partial Reliability for MPEG-4 Multimedia Streaming." *Proc. of BSDCon Europe* 2002.