IEEE Int. Conf. on Multimedia & Expo, Baltimore, July 2003

BETTER AUDIO PERFORMANCE WHEN VIDEO STREAM IS MONITORED BY TCP CONGESTION CONTROL

Longin Jan Latecki *latecki@temple.edu*

Kishore Kulkarni *kkulkarn@temple.edu*

Jaiwant Mulik *jmulik@temple.edu*

Dept. of Computer and Information Sciences, Temple University
Philadelphia, PA 19122, USA, Tel. +1 215 204 5781

ABSTRACT

Conventional wisdom holds that the TCP like congestion control is unsuitable for real-time multimedia conferencing. However, our results clearly show that an audio and video conferencing system that transmits video over TCP (and audio over RTP/UDP) can provide significantly better audio quality to the end user than one built on RTP/UDP alone. We measured audio quality in terms of packet loss, packets arriving too late (for real time play out), average packet delay, and jitter. Our results also clearly indicate that sending video over TCP does not introduce any additional delay in the arrival time of video packets in comparison to RTP/UDP.

1. INTRODUCTION

There is consensus among Internet architects that end-to-end congestion control is fundamental to the health of the Internet. Fall and Floyd [2] note that “In the current architecture, there are no concrete incentives for individual users to use end to-end congestion control, and there are, in some cases, “rewards” for users that do *not* use it (i.e. they might receive a larger fraction of the link bandwidth than they would otherwise).” They go on to argue that additional incentives must be built into the network to encourage the use of TCP-friendly protocols, that is, those that reduce their load on the network when they experience packet loss (indicating congestion). Such incentives may include social incentives (i.e., avoiding the embarrassment of having your protocol/software labeled non-TCP friendly) pricing incentives, and traffic policing (providing degraded service to non-responsive flows).

Underlying all of this is the assumption that users, left to their own devices, would just as soon blast as many packets over the network using UDP as they could get away with, and that TCP congestion control, like a speed limits on a highway, is a regulation that society recognizes is necessary, but individuals gain from disregarding.

In this paper, we argue that TCP congestion control is not always your enemy. In fact, using TCP rather than UDP for video transport can actually improve the performance of audio stream when the current available bandwidth of a network path is unknown (which is the normal case in the Internet) and that available bandwidth is less than the maximum data rate of the video stream.

Clearly, TCP congestion control can introduce the play out delay of video packets due to the retransmission. However, the influence of this delay on video quality can be controlled at the application layer. On the other hand, our results indicate that when video is transmitted over RTP/UDP, less than 1% of frames arrive without any packet loss even by minor packet loss conditions. This means that more than 99% of the received video frames would possibly show visual artifacts in played video.

In this paper, we define a particular problem in network design that we call the “two-stream” problem. Our formulation of the two-stream problem is motivated by applications like video conferencing with the simultaneous transmission of real time audio and video, but is not limited to multimedia. Our long-term research goal is to investigate various combinations of application and transport layer protocols to address this two-stream problem. This short paper has a more modest goal: to illustrate that sending real-time video over TCP is not only good for the network, but good for the end-user as well, in the case of simultaneous streaming of audio and video.

Our main claim is that a TCP-friendly congestion control applied to video stream substantially improves the quality of audio stream. Note that we do not claim that TCP is optimal for sending video. As we point out, the retransmissions inherent in TCP introduce a tradeoff between reliability and delay. Alternative protocol designs that incorporate TCP-friendly rate control into an unreliable message oriented protocol (e.g., DCCP [4] and other related work) may provide a better long-term solution. Our purpose is rather to refute the conventional wisdom that TCP is a bad fit for real time video with data that suggest otherwise, and hopefully to encourage a more favorable view towards the effects of TCP-friendly congestion control on streaming media in general.

1.1 The Two-Stream Problem:

Consider a video telephony application with two streams to send, where the audio stream (A) is of high priority with a low fixed bit rate, and the video stream (V) is of lower priority. In video conferencing applications it is well known that users would rather accept degradation in video than in audio quality. The sender transmits streams A and V to a receiver over IP. In stream A, packets are sent regularly, for example, every 50 ms. Assuming that the available transmission bandwidth is sufficient to successfully transmit stream A, we want to also transmit stream V with as much bandwidth as possible without affecting the quality of stream A. We also want a minimal delay for both streams.

Any solution to the two-stream problem needs to consider the transport protocols used for both streams and the interaction with the application. Since transmission of stream A has higher priority than transmission of stream V, the performance of stream A is our main criterion to measure the quality of the solution. In this paper we present experimental results that demonstrate that transport of stream V over TCP provides a significantly better solution to the two-stream problem than transport over UDP. Under the same network conditions, both the number of dropped packets and the number delayed packets are significantly better for stream A when stream V is transported over TCP.

2. EXPERIMENTAL SETUP

We conducted experiments using our audio-video clients in three different network environments:

1. Emulab - the Utah Network Emulation Testbed (Netbed) [7],
2. Our network with *NistNet* emulator [6], and
3. Dial-up connection with 28.8 kbps over the Internet.

In all our experiments stream A is an audio stream that is sent as RTP over UDP. We send every 50 ms an audio packet of 96 Byte size. Since we transport 20 packets per second, we have 1,920 Byte/s of audio data. Our video stream V is also sent in RTP packets. In order to have the same conditions in all experiments, we streamed video from a prerecorded video file that produces a video stream. In order to preserve the original video quality, this stream requires the transmission of 264.436 Kbps (33,054 Byte/s) for the total time of 76 seconds.

All experiments followed the same scenario:

1. We first streamed just audio data for 60 seconds,
2. Then we added our prerecorded video stream for 76 seconds,
3. Finally again we sent only audio data for the last 60 seconds.

We performed two runs of experiments, in one run stream V was sent over UDP, and in the second run stream V was sent over TCP. The statistics are obtained from our audio-video software client. All times are recorded at the time audio and video packets were given to the transport layer or were received form the transport layer.

When there were no bandwidth restrictions and no packet drop, we obtained the optimal video throughput of 264.436 Kbps in 76 seconds. The situation for arrival of audio packets for both transport protocols UDP and TCP was the same. The average interarrival time of audio packets was as expected (50 ms). The standard deviation of the interarrival time (std) was zero at the times when only audio was transmitted and it increased slightly to about 5 ms when audio and video were transmitted together.

3. EXPERIMENTAL RESULTS

We performed several experiments under the experiment settings described in the last section. In these experiments we varied the bandwidth and the packet loss. Our experiments clearly demonstrate that when video is sent over TCP not only the packet loss is smaller but also the delay of audio packets is significantly smaller. Since the results of all our experiment show the same phenomena, we describe one representative experiment in detail in this section. We set the following parameters using NistNet:

- 15500 Bytes/s bandwidth limit and
- 5% packets drop rate.

3.1 Interarrival time of audio packets for video over UDP

The sender sent 3932 UDP audio packets, 3710 packets arrived, and consequently, 222 audio packets were dropped. This is equivalent to 11 seconds of audio data loss. In addition 220 audio packets arrived more than 250 ms after the arrival of the previous packet. The expected time of arrival for an audio packet is 50 ms after the arrival of the previous audio packet. Therefore, any packet that arrives >= 250 ms after the arrival of the previous packet is at least 200 ms late. Hence, additional 11 seconds of audio data would be lost due to play out buffer overflow when the play out jitter buffer for audio is set to 200 ms. This means that the end user would lose 22 seconds audio.

Since the results of all our experiments are consistent, we present a detailed report on our experiments with NistNet and only a brief overview of the experiments in the other two network environments. Our software and experiment reports can be found on www.cis.temple.edu/~latecki/TwoStream.

We have two machines running our audio-video clients, and one machine with *NistNet* emulator [6] (version 2.0.11) in between. All three computers are Celeron 366 MHz PCs and have RedHat Linux 7.2 operating system. They are in an isolated network connected using 10Mbps Ethernet.

NistNet emulator is a Linux kernel-level module that can be used to emulate performance dynamics in IP networks, by allowing settings of various network characteristics like bandwidth, delay, and queuing parameters. *NistNet* uses the *Derivative Random Drop* (DRD) congestion control algorithm (Gaynor [3]) and allows configuration of the minimum and maximum queue length. The queue length in measured in the number of packets. When the queue length reaches the configured minimum queue length, DRD starts dropping 10% of packets and the loss percentage continually increases until the actual queue length reaches the configured maximum queue length. At the configured maximum queue length, DRD loses 95% of packets.

3.2 Interarrival time of audio packets for video over TCP

The sender sent 3932 UDP audio packets, 3762 packets arrived, and consequently, 172 audio packets were dropped. Observe that 52 less packets were dropped in this case. Thus, the packet loss is about 23% smaller. This is equivalent to 2.5 seconds of more audio data than in the case of UDP. Moreover, only 7 packets arrived more than 250 ms after the arrival of the previous packet. This is a huge difference in favor of TCP in comparison to video transfer over UDP. The total end user loss of audio data is twice smaller for TCP with the 200 ms play out jitter buffer for audio. It is reduced to 8.8 seconds for TCP. The situation is similar for other values of the audio play out jitter buffer. This can be clearly seen by comparison of Figures 1 and 2. For example, there are still 220 audio packets that arrived more than 500 ms after the arrival of the previous packet when video is transmitted over UDP (Fig. 1).

3.3 Video packets between audio packets

To show how video transmission influences the audio transmission we measured the average interarrival time for audio packets that contain at least one video packet in between them. This value was 756 ms when video was sent over UDP, whereas average interarrival time for audio packets with no video packet in between them was 21ms. Average of interarrival time for audio packets with video in between them was 122 ms when video was sent over TCP, whereas average interarrival time for audio packets with video in between them is 43ms. These numbers indicate that video transport over TCP caused significantly less delay to the interarrival time (decreased interarrival time for audio packets with video packets between them) as well as significantly less burst (the interarrival time closer to the ideal value of 50 ms was obtained for consecutive audio packets with no video packets between them).

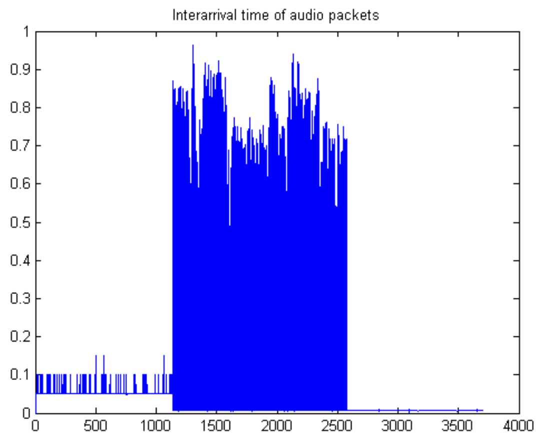

Figure 1. Interarrival time of audio packets, video over UDP. X axis: packet sequence number, Y axis: time in seconds.

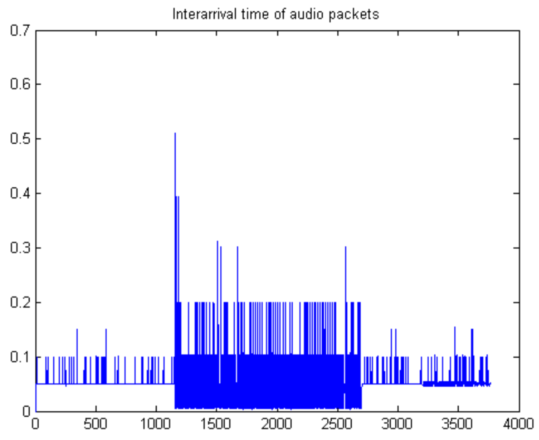

Figure 2. Interarrival time of audio packets, video over TCP. X axis: packet sequence number, Y axis: time in seconds.

3.4 STD of audio packets

Further advantages of sending video over TCP can be obtained by comparison of Figures 3 and 4 that show values of the standard deviation (std) of the interarrival time taken over last 100 packets. Whereas the maximal value of the std for audio packets during the video transmission over TCP is below 100 ms (Fig. 4), it becomes more than 300 ms when video is sent over UDP in Fig. 3. Moreover, the mean std for audio packets during the video transmission over TCP is about 60 ms (Fig. 4), whereas it is larger than 250 ms during the video transmission over UDP (Fig. 3).

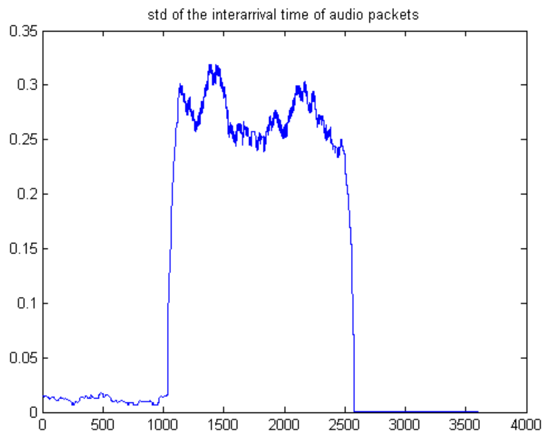

Figure 3. Std of the interarrival time of audio packets, video over UDP. X axis: packet sequence number, Y axis: time in seconds.

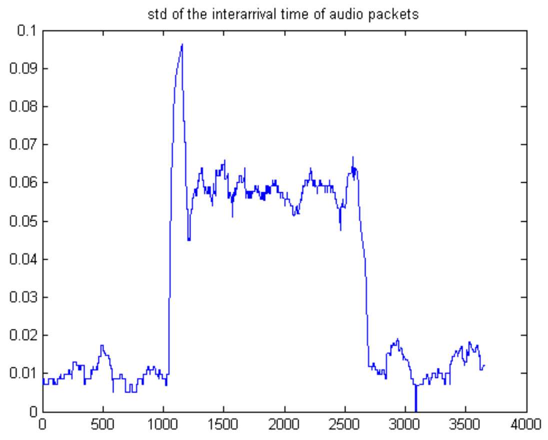

Figure 4. Std of the interarrival time of audio packets, video over TCP. X axis: packet sequence number, Y axis: time in seconds.

3.5 Results for video packets

Clearly, one has to pay the price for the improved audio performance. As expected less video data can be transmitted over TCP, but the difference is surprisingly small. Video data over UDP arrived with the average speed of 110.801 Kbps, whereas for TCP it was 102.676 Kbps. Moreover, the delay of video packets that arrived over TCP was smaller than over UDP.

4. OTHER NETWORK SETTINGS

We also performed experiments in Emulab and in real network conditions and found similar results. We present a brief summary of the results of a representative experiment in each case.

Emulab(Netbed [7]): The setup consists of two RedHat Linux 7.1 machines with a FreeBSD 4.5 machine in between that runs dummynet to control the bandwidth, which is set to 15500 bytes/s with 5% loss.

The real network: We established a 28.8 Kbps dial-up connection between a sender (Pentium 1.6 GHz, RedHat Linux 7.3, 10 Mbps Ethernet) and receiver (Pentium 236 MHz, RedHat 7.2, 28.8 Kbps dial-up).

Experimental results are summarized in the following table, where (1) is the total amount of audio data lost for the play out due either to the packet loss or late arrival (for play out jitter buffer of 200ms), (2) is the avg. inter-arrival time for audio packets with video packets in between, (3) is the mean std. of audio packets when video is also being streamed along, (4) is the video bit rate at the receiver, (5) represents frames sent / frames received without packet loss. The second row indicates the transport protocol used for video transmission.

	Emulab [Netbed]		Real Network (28.8 Kbps)	
	TCP	UDP	TCP	UDP
1	11.5 sec	20 sec	27 sec	55 sec
2	190 ms	360 ms	635 ms	537 ms
3	65 ms	115 ms	150 ms	350 ms
4	96.34 Kbps	93.18 Kbps	20.12 Kbps	23.32 %
5	82/82	209/1	30/30	220/7

Clearly, the data in the table indicate significantly better audio performance when video is sent over TCP. This is particularly obvious in (1), (3), and (5). Only the value in (2) for real network is slightly higher for TCP then for UDP, but the values are comparable. The values for (4) are comparable, which means we receive nearly the same amount of video. However, nearly no video frame arrived without any packet loss over UDP (5).

5. VIDEO PERFORMANCE

As we stated in the introduction, TCP congestion control can introduce the play out delay of video packets due to the retransmission. However, in all our experiments the play out delay was in average less than half of the interarrival time of video frames. Moreover, the influence of this delay on video quality can be controlled at the application layer. On the other hand, when video is transmitted over RTP/UDP, less than 1% of frames arrive without packet loss even with a minimal network packet loss conditions. For example, in our experiments on Emulab, in average only one frame out of 209 sent frames arrived without any packet loss when video was sent over UDP. Due to TCP congestion control only 82 frames were sent and arrived (without any packet loss). Average interarrival time of the frames was one second, and average delay was 400 ms.

To summarize, transmission of video over TCP may lead to additional frame freezing, but transmission of video over UDP definitely introduces packet loss when the bandwidth drops. Several years of research in video compression did not provide any acceptable solution to compensate for video quality degradation due to the packet loss. In our opinion, it is much easier task to reduce the frame freezing effect. Possible solutions on the video client level include: sending the most recent frame, and on the transport level: restriction of the retransmission time of video packets and transmission of consecutive frames in different streams, e.g., by using partial reliable SCTP as proposed for MPEG 4 stream in [5] or using DCCP.

6. RELATED WORK

An alternative to TCP is to use the *Congestion Manager* (*CM*) described in [1]. The CM provides a mechanism for sharing congestion information across multiple flows. With the *CM* in place, a videoconferencing could use the *rate callback* method of data transmission. The rate callback mechanism is well suited for videoconferencing applications that transmit at a fixed schedule. An application using the rate callback method registers a callback that is invoked by the CM every time the allowable rate falls below or goes above the configured thresholds. A video conferencing application could use this feedback from the congestion manager to adjust the video stream. The basic idea is that the CM allows sharing congestion information between flows. This information can be used to eliminate the two-stream problem that occurs either due to a slow host or/and due to a network bottleneck.

7. ACKNOWLEDMENTS

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.

8. REFERENCES

[1] D. Andersen, D. Bansal, D. Curtis, S. Seshan and H. Balakrishnan. System support for bandwidth management and content adaptation in Internet applications. *Proc. Symposium on Operating Systems Design and Implementation*, pp. 213-226, San Diego, CA, October 2000.

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

[3] M. Gaynor, Proactive Packet Dropping Methods for TCP Gateways, 1996. http://www.eecs.harvard.edu/~gaynor/final.ps

[4] E. Kohler, M. Handley, S. Floyd, and J. Padhye. Datagram Congestion Control Protocol (DCCP). *Internet Engineering Task Force, INTERNET-DRAFT*, http://www.ietf.org/internet-drafts/draft-ietf-dccp-spec-00.txt

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

[6] NistNet, software provided by National Institute of Standards and Technology, http://snad.ncsl.nist.gov/itg/nistnet/

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