Characterizing the Behaviour of RealVideo Streams

Sean Boyden Anirban Mahanti Carey Williamson

Department of Computer Science, University of Calgary, 2500 University Drive NW, Calgary, AB, Canada T2N 1N4
Email: {boyden, mahanti, carey}@cpsc.ucalgary.ca

***Abstract*— Unresponsive streaming media traffic can significantly impede the performance of other networking applications and even endanger the stability of the Internet. Therefore, with increased deployment of high-bandwidth commercial streaming video applications on the Internet, understanding the characteristics of proprietary streaming formats is pertinent. In this paper, we use an experimental testbed to characterize the behaviour of RealVideo streams under different network conditions, with special emphasis on understanding their "TCP friendliness". In many scenarios, our experimental results show that RealVideo streams are not TCP friendly. The results also show that the new TurboPlay feature fundamentally changes the behaviour of RealVideo streams, making them more aggressive.**

Keywords: RealVideo; Media streaming performance; Network measurement; Network emulation; TCP friendliness

I. Introduction

Recent years have witnessed ever-increasing demand for media-on-demand applications on the Internet. Typically, users access online media clips by clicking on a hyperlink using their Web browser, which results in the browser opening a media player to play the selected media file. Usually, the server delivers the selected media file to the player using a technique known as *streaming*. With streaming, there is a brief buffer-filling period for the initial portion of the media, and then the remainder of the media is obtained across the network as the media is being played.

Digitally encoded media consists of a sequence of audio and video frames with precise timing relationships that must be maintained as faithfully as possible during playback. Streaming applications require sustained network bandwidth; the media transmission rate must be no less than the rate at which the media player consumes data.

Streaming media data can be delivered using either the Transmission Control Protocol (TCP) or the User Datagram Protocol (UDP) at the transport layer. However, TCP employs congestion control schemes that adapt dynamically to network conditions, often yielding variable transmission rates and packet delays. For this reason, many streaming media applications use UDP as the transport protocol.

Regardless of the transport-layer protocol used, it is important to consider how media streaming flows interact with other network applications. The performance of the Internet depends largely on the flow and congestion control capabilities built into TCP, which carries a vast majority of the traffic [5]. Depending on media quality, streaming bandwidth requirements can range from 56 kbps for modem users to 30 Mbps for HDTV-quality video. Thus, unresponsive streaming media traffic can potentially impede the performance of other applications that employ TCP, or worse, endanger the stability of the Internet.

These considerations have prompted interest in *TCP friendly* rate control protocols [6], [13]. The TCP friendly paradigm recommends that the bandwidth usage of a UDP-based multimedia flow should not, on average, exceed that of a TCP flow under similar conditions [2]. Fairness concerns have also motivated workload characterization studies of media streaming applications [3], [9], [15], [16], and considerable interest in the performance evaluation of commercial media streaming products [4], [7], [8], [10], [17].

Since the growth in popularity of streaming media applications continues, the degree of their TCP-friendliness will have a non-trivial impact on the stability and performance of the Internet. While many TCP friendly rate control protocols have been proposed, it is not clear if these, or related protocols, have been incorporated into commercial streaming media products. While there are some indications that UDP-based streaming applications adjust the media streaming bit rate based on available bandwidth [4], [10], [17], little is known about the behavioural characteristics of these proprietary protocols.

The purpose of this paper is to characterize the behaviour of RealVideo media streaming applications. Our work is carried out experimentally, using a network emulator to vary the network bottleneck bandwidth and cross-traffic conditions between server and client. Our study focuses on the qualitative and quantitative behaviour of RealVideo streams, as well as on their interactions with TCP flows.

Our results illustrate three important behavioural properties of RealVideo streams. First, the buffering behaviour of the client varies according to the encoding rate of the media stream. For lower-bandwidth streams, the buffering bit rate is 2-4 times higher than the encoded rate, to a maximum of approximately 600 kbps. For higher-bandwidth streams, the buffering takes place at the encoding rate. Second, we studied the "TurboPlay" option in RealNetwork's media client. The TurboPlay option is intended to reduce the buffering time for media playback. Our experiments show that with TurboPlay option enabled, the streaming rate to the client is initially very high (almost equal to the available bottleneck link bandwidth), regardless of the encoding rate or the presence of competing traffic. With TurboPlay enabled, RealVideo Streams are not TCP friendly. With the TurboPlay option disabled, the TCP friendliness of RealVideo streams may increase.

Finally, we observe that *SureStream* technology provides some compromises. It can dynamically reduce the media bit rate to allow some TCP throughput, but is still not truly TCP friendly.

The remainder of this paper is organized as follows. Section II provides a brief discussion of related work on the characterization of media streaming applications. Section III describes the experimental methodology for our study. Section IV presents the results from our study. Section V concludes the paper and outlines our plans for future work.

II. Background and Related Work

RealSystem [11] is the Internet solution for audio and video streaming proposed by RealNetworks. They provide tools such as RealServer, RealPlayer, and RealProducer, corresponding to the media server, the client, and codec, respectively. The RealSystem architecture supports both real-time and on-demand streaming. We only study on-demand streaming in this paper.

The RealAudio and RealVideo contents are created in advance using RealProducer, and stored in RealMedia File Format (RMFF [1]) files. Before encoding, a target bit rate is chosen, based on the video quality desired for the intended audience. The bandwidth for the audio stream is allocated first, then that for the video stream. One way that RealVideo achieves compression is by skipping frames when needed, so as to achieve a high frame rate for action scenes, and a low frame rate for low-activity scenes.

To enable dynamic and seamless bandwidth adjustments, media is often encoded at multiple bit rates but stored as a single media object. When a client requests this media object, the server ascertains which encoding best suits the bandwidth available to the client. If the transmission path between the client and server changes during playback, the server can dynamically choose another bit rate that best matches the current network conditions.

This application-layer bandwidth adjustment technique is available in several commercial products. For example, RealNetworks calls this technology "SureStream" [17], while Microsoft calls it "Intelligent Streaming" [10]. If only a single bit rate is available, responding to congestion would entail selectively dropping frames. This is also the case if the client has reached the lowest bit rate offered in a combined media object. For example, a server may respond to unfavourable network conditions by first decreasing the frame rate of the video, and if insufficient, it may (even) drop all video frames and only transmit audio.

Prior work on performance of commercial streaming applications has evaluated RealVideo streaming in wireless environments [7], studied congestion responsiveness of RealPlayer [4], [17], and compared RealPlayer and Windows Media Player performance [8]. Although prior work has studied the congestion responsiveness of RealVideo streams, most have relied on accessing content from the Internet. Our work is carried out experimentally using a network emulator. This approach allows control of all parts of the utilized network, including the clients, the servers, the cross traffic, as well as the media content. Furthermore, we focus on the qualitative and quantitative behaviour of RealVideo streams, as well as on their interactions with TCP flows. This research complements recent work by Nichols *et al.* [10] that characterized how Windows Media responds to congestion by utilizing a similar experimental testbed.

III. Experimental Methodology

Our experiments were conducted on an experimental testbed in our laboratory. This testbed consists of five physical machines connected over a 10 Mbps private Ethernet LAN. In our tests, two machines act as servers, two machines act as clients, and one machine acts as a network emulator. The network emulator was used to model an arbitrary internetwork between the client and server machines.

A. Testbed Configuration

In the testbed, the streaming media server was run on a 2.4 GHz Intel Xeon machine with 1 GB of RAM. The media client was run on a 2 GHz Intel Pentium 4m machine with 256 MB of RAM. We used RealNetworks Helix DNA Server Basic[1] (version 10) as the media server. The media client in the experiments was the RealPlayer[2] (version 10), which is based upon the open-source HelixPlayer.

TCP cross-traffic was generated using `netperf`[3]. This software is freely available from Hewlett-Packard. It is a tool for benchmarking network performance, with a primary focus on bulk data transfer over TCP or UDP. By selecting TCP_STREAM mode, network traffic representing a long-duration FTP flow can be generated. A 2.4 GHz Intel Xeon machine with 1 GB of RAM was used to run the `netperf` server. The `netperf` client was running on a machine with an identical configuration.

Our work uses the Internet Protocol and Traffic Network Emulator (IP-TNE), a high-performance internetwork emulator that provides a detailed simulation model of arbitrary internetworks [14]. Using this emulator, hosts can send IP packets to other hosts, whether real or virtual (simulated) via the emulator. The conversion of packets between real and simulated network environments is accomplished through a technique similar to IP masquerading. IP-TNE was run on a 2 GHz Intel Xeon dual-processor machine with 4 GB of RAM.

All nodes employ Linux 2.6.8 as the operating system, except for the media server node which runs Linux 2.4.2.

B. Emulated Network

The experiments whose results are reported here emulate a simple dumbbell network topology with a common bottleneck link between all sources and sinks. Each source/sink is connected to the bottleneck link by a high bandwidth access link. For all experiments, the bottleneck capacity was set to one of four settings: 250 kbps, 750 kbps, 1500 kbps, and 100 Mbps. The first three levels represent typical speeds for home Internet access. The 100 Mbps setting is used to represent "unlimited"

[1]http://www.realnetworks.com/products/discreteserver/index.html
[2]https://player.helixcommunity.org/
[3]ftp://ftp.cup.hp.com/dist/networking/benchmarks/netperf/

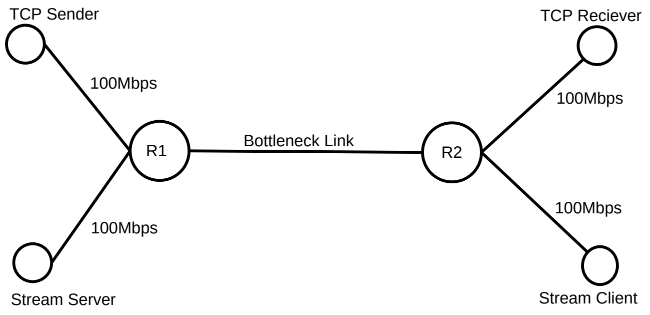


Fig. 1. Emulated network configuration

network capacity. Figure 1 provides a graphical description of the emulated network configuration.

The round trip propagation delay on the emulated network is 105 ms which is representative of typical WAN delays on the Internet. This round trip delay value includes a 50 ms one-way delay on the bottleneck link and about 2 ms delays on the emulated end networks. Note that the actual bandwidth available to these end hosts is 10 Mbps, as that is the capacity of the physical network.

The routers in the emulated network use FIFO queueing with drop-tail queue management. In our experiments, the bottleneck router can queue up to 51,200 bytes.

C. Experimental Design and Performance Metrics

Our experiments fall into two categories: those with no TCP cross traffic, and those with TCP cross traffic. The former set of experiments focuses on the dynamics of RealVideo streams, while the latter set focuses on TCP friendliness.

The first set of experiments (no TCP cross traffic) used a combination of application-layer and network-layer performance measurements. This measurement approach provides multiple perspectives for data analysis, and enables cross-validation of our results.

At the application layer, the RealPlayer client statistics were recorded to a file every 0.5 seconds. This method of data collection required modification to the statistics tracker in the client. Other adjustments to the RealPlayer involved inserting text markers in the statistics file to mark the beginning and end of media playback. These markers were important for the subsequent analysis of these data files, especially in determining the duration of the buffering period.

The RealPlayer statistics were written to a RAM-disk in order to minimize the delays for logging. From the client statistics, we can determine characteristics such as encoded bandwidth, actual network bandwidth, frame rate, packet losses, packet retransmissions, late packets, and more.

At the network layer, we used `tcpdump`[4] at the client side to check the accuracy of the RealPlayer statistics. One UDP trace was created per experiment. The collected information was recorded to a RAM-disk, similar to the RealPlayer statistics. After the playback of the selected media was complete, the UDP trace was copied from the RAM-disk to the physical disk for analysis. From this trace we can determine characteristics of the stream packets at the network layer, including packet type, packet size, and timestamp.

For the second set of experiments (with TCP cross traffic), an additional `tcpdump` trace of TCP traffic was recorded as well. The trace was collected on the `netperf` server machine using the same RAM-disk method as for the UDP trace. From this TCP trace we can determine information such as packet type, packet size, TCP flags, timestamps, and more.

D. RealVideo Streams

The streams used in these experiments were produced with RealProducer Plus 10.0 (based on Helix DNA Producer)[5]. This tool takes a source video and encodes it in RealMedia File Format [1]. The codec utilized by default with this software is RealVideo (RV) 10, for which an overview is provided in a white paper from RealNetworks [12]. RealVideo can be encoded in either CBR or VBR. In this work, we focus on CBR streams; we plan to study VBR streams in future work.

RealProducer encodes video streams according to different target "audiences". The "audience" setting allows one, for instance, to create a stream with characteristics suitable for a user on a 56 kbps dialup link. The actual bit rate of the generated video stream is typically lower than the target audience rate, leaving some bandwidth for control information or other user tasks. The CBR streams used in this paper are encoded at the following "audience" rates: 56 kbps, 128 kbps, 256 kbps, 384 kbps, 512 kbps, 768 kbps, 1000 kbps, and 1500 kbps. The "audience" rate, however, is not the true bit rate of the stream. The true bit rates are 34 kbps, 90 kbps, 230 kbps, 350 kbps, 730 kbps, 1000 kbps[6], and 1400 kbps, respectively. In the rest of the paper, the stream bit rate refers to the "audience" bit rate, unless explicitly stated otherwise.

RealProducer, in addition to single-rate streams, allows the creation of SureStream (SS) streams with multiple bit rates. Content encoded using SureStream is more amenable to dynamic rate adaptation. One SureStream stream was created from all of the foregoing CBR streams; this stream is used to study the TCP friendliness of SureStream streaming[7].

IV. Results

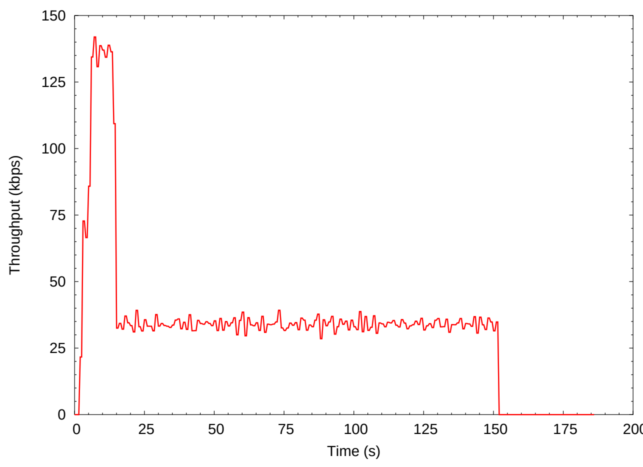
This section presents the results from our two sets of experiments. First, the behaviour of RealVideo is characterized in the absence of cross traffic. These baseline experiments focus on the buffering behaviour of the media client, the effect of the encoded media bit rate, and the impact of the TurboPlay option. Second, the behaviour of RealVideo streams is analyzed in the presence of TCP cross traffic. These experiments are used to evaluate the TCP friendliness

[4]http://www.tcpdump.org

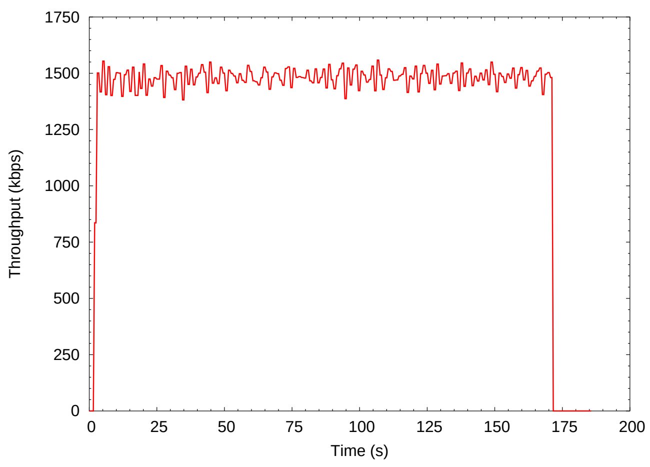
[5]http://www.realnetworks.com/products/producer/

[6]For the 1000 kbps stream, the true bit rate is also 1000 kbps. This was a configuration error when encoding the stream. This error has minimal impact on our results.

[7]The 7-minute SureStream represents a full-length encoding of the original AVI. The shorter videos used in other experiments were encoded from a clipped version of the original.



(a) Low bit rate stream (56 kbps)



(b) High bit rate stream (1500 kbps)

Fig. 2. Streaming bandwidth consumption over unlimited bottleneck link (TurboPlay disabled)

TABLE I
SUMMARY OF RESULTS FOR UNLIMITED BOTTLENECK CAPACITY (TURBOPLAY DISABLED)

Encoded Bit Rate	Buffering Period (s)	Avg. Buffering Bit Rate (kbps)	Avg. Playback Bit Rate (kbps)	Peak Bit Rate (kbps)
56	6.02	45	32	141
128	7.00	133	94	416
256	5.51	281	217	628
384	7.51	379	336	636
512	7.00	366	432	643
768	6.05	414	680	918
1000	5.50	562	976	1144
1500	5.51	817	1362	1558
SS	6.51	851	1336	1556

of the media streams. The results for the first and second set of experiments appear in Section IV-A and Section IV-B, respectively.

A. Baseline Experiments (No Cross Traffic)

The first experiment studies the RealPlayer behaviour in a simple scenario with unlimited bottleneck capacity and the TurboPlay option disabled. We initially focus on the buffering behaviour before playback begins.

Table I summarizes the measurement results from this experiment. The table shows the buffering duration, the mean bit rate during buffering, the mean bit rate during playback, and the peak bit rate for different encoded streams over the unlimited bottleneck link. The buffering period represents the elapsed time between receiving the first packet from the server and starting video playback. This time represents the filling of the buffer before playback commences. Several observations can be made from this summary of results. First, all streams have a buffering period of 5-7 seconds, regardless of their encoded bit rate. Second, for streams encoded at low bit rates (i.e., 56 kbps to 384 kbps), the mean bit rate during the buffering period is higher than the mean bit rate during the playback period. Third, the mean bit rate during playback of low bit rate streams tends to be about 80% of the encoded bit rate, while the peak bit rate can be as much as 2-4 times higher than this.

Figure 2 shows a time series representation of the observed bit rates for media streaming. This figure illustrates the qualitatively different behaviours observed for streaming media encoded at low or high bit rates. For streams encoded with a low bit rate, there is a distinct initial period of higher bandwidth consumption, in which the client receives data at approximately 2-4 times the encoded bit rate. This behaviour is illustrated in Figure 2(a) for the 56 kbps stream. The initial portion shows observed bit rates near 140 kbps. The rest of the stream uses about 32 kbps. For the media streams encoded at higher bit rates (i.e., 512 kbps, 768 kbps, 1000 kbps, and 1500 kps), this dual-rate behaviour is not observed (also see Table I). While the client still does buffering before initiating playback, the buffering rate is indistinguishable from the streaming bit rate, as illustrated in Figure 2(b).

The next experiment considers the impact of the bottleneck link capacity on the streaming behaviour. We use the IP-TNE network emulator to change the emulated network capacity between the streaming server and the RealPlayer client.

When the bottleneck bandwidth is close to the encoded media bit rate, the stream still achieves its target bandwidth. This behaviour is illustrated in Figure 3 for the 256 kbps media stream using a 250 kbps bottleneck link. Similar observations apply for the 768 kbps stream on the 750 kbps bottleneck, and the 1500 kbps stream on the 1500 kbps bottleneck.

Streaming still works well in these scenarios because the actual bit rate is less than the bottleneck link capacity. However, at close to capacity we observe increased packet loss in the stream. Figure 4 presents the cumulative packet loss for the three scenarios mentioned. Steady packet loss is observed when the media encoding rate and bottleneck capacity are similar. The loss is most notable for the 768 kbps and 1500 kbps streams.

When the bit rate of the stream exceeds the bottleneck link capacity, two distinct phenomena are observed. For the

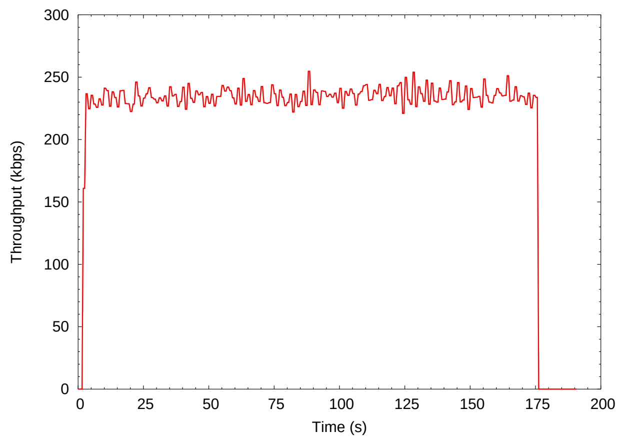


Fig. 3. Streaming bandwidth consumption over constrained bottleneck link (256 kbps media/250 kbps bottleneck, TurboPlay disabled)

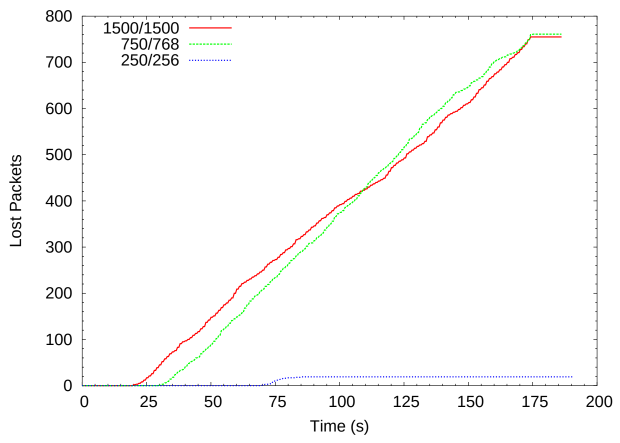
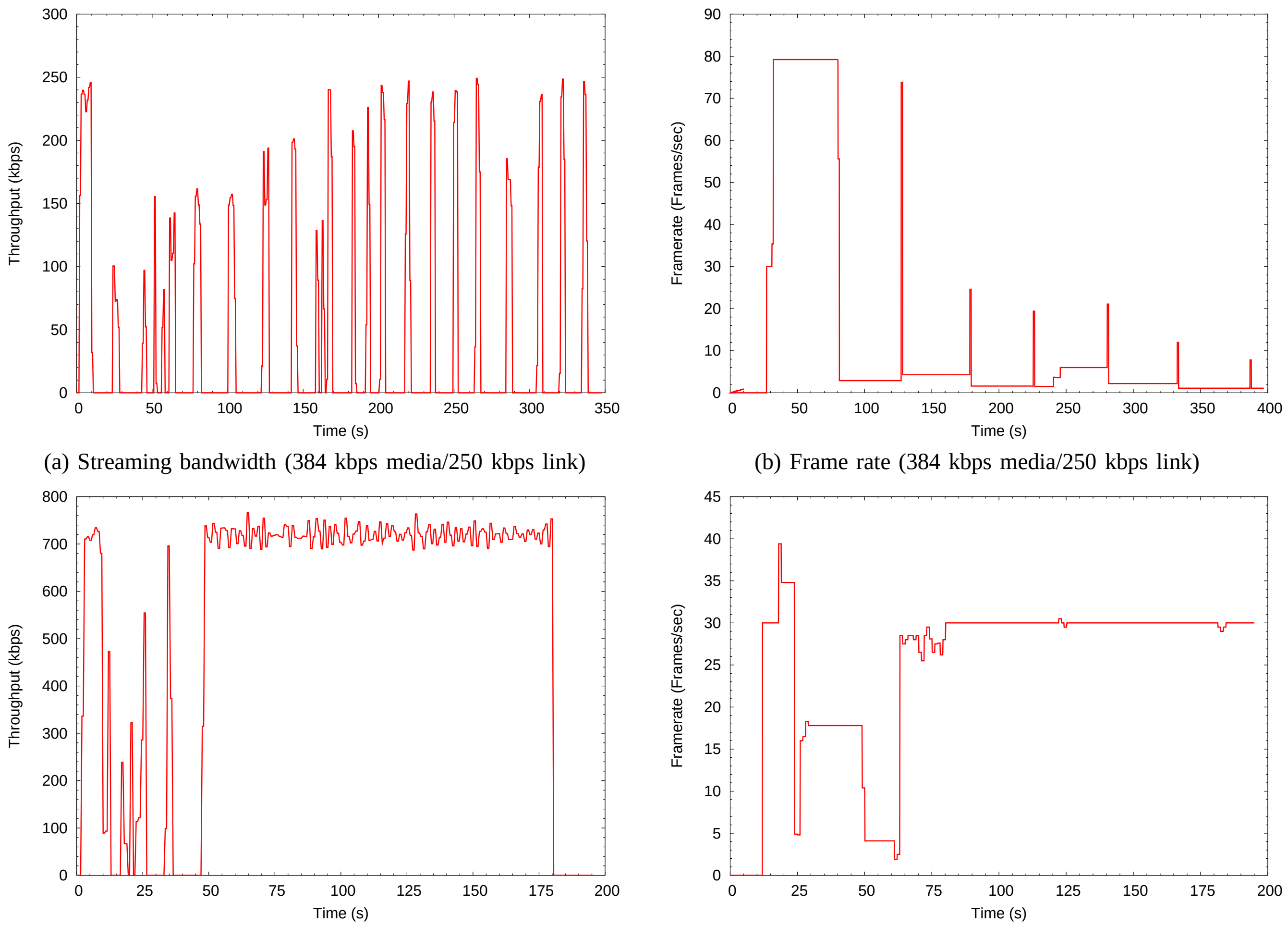


Fig. 4. Cumulative packet loss results for media streams over bottleneck link close to encoded media bit rate (256 kbps media/250 kbps bottleneck, 768 kbps media/750 kbps bottleneck, 1500 kbps media/1500 kbps bottleneck, TurboPlay disabled)



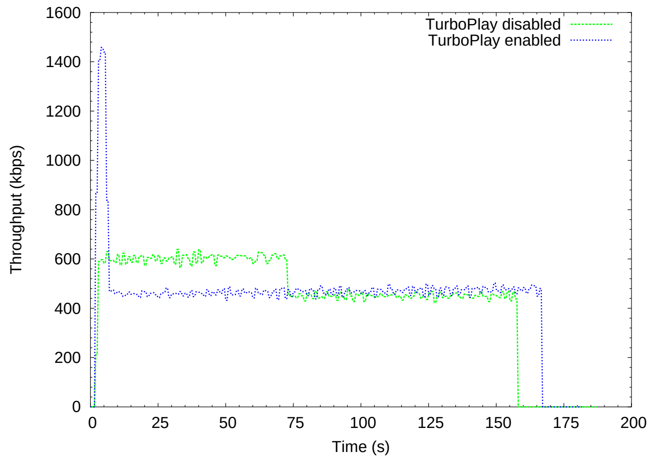
(a) Streaming bandwidth (384 kbps media/250 kbps link)

(b) Frame rate (384 kbps media/250 kbps link)

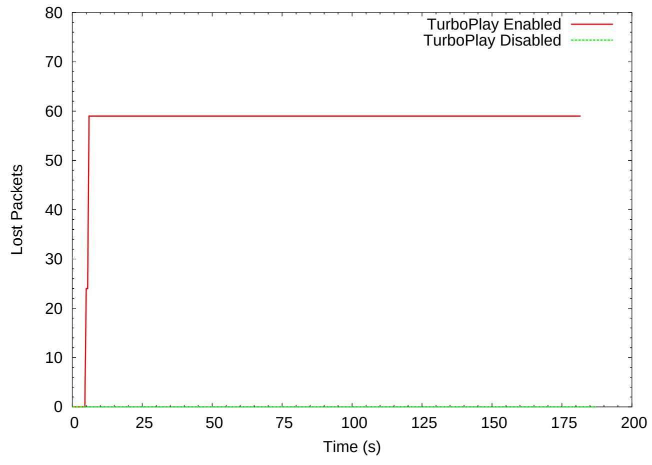
(c) Streaming bandwidth (1000 kbp media/750 kbps link)

(d) Frame rate (1000 kbps media/750 kbps link)

Fig. 5. Streaming behaviour over an under-provisioned bottleneck link (TurboPlay disabled)



(a) Bandwidth usage



(b) Cumulative packet loss

Fig. 6. Effect of TurboPlay on the bandwidth consumption of a 512 kbps stream over a 1500 kbps bottleneck link

low-bit-rate media streams, the client application "gives up" on retrieving video frames, and reverts to audio only. For higher-bit-rate media streams, the streaming continues, but with reduced quality.

The behaviour for low-rate streams is illustrated in Figures 5(a) and (b), for the 384 kbps stream over a 250 kbps bottleneck. The throughput of the stream is erratic (Figure 5(a)), as is the frame rate (Figure 5(b)). The spikes represent audio data received from the server periodically.

The behaviour for high-rate streams is illustrated in Figures 5(c) and (d), for the 1000 kbps stream over a 750 kbps bottleneck. In these graphs, the client first buffers some data, starts playback, and then appears to encounter difficulties. The limited bottleneck capacity constrains the throughput (Figure 5(c)), as well as the frame rate (Figure 5(d)). The surprising result is that the player continues to play the video in this situation, at nearly 30 fps, but with noticeable quality loss. To some users this may appear unacceptable. Near-capacity utilization of the bottleneck is observed after the initial instability.

Based upon the throughput/frame rate results and the playback quality, we hypothesize that the higher-bit-rate streams contain multiple components contributing to the final video quality, and that the player is able to play an appropriate subset of these under poor network conditions. The lower-bit-rate media streams must not have this feature.

The next experiment studies the TurboPlay client option. This option is a relatively recent feature for RealMedia streaming. RealNetworks claims that this option provides a better streaming experience for users with broadband connections. In particular, this feature reduces the user-perceived startup delay for video streaming.

Figure 6 shows selected results from our experiments with the TurboPlay feature. This experiment studies the transmission of the 512 kbps stream over a 1500 kbps bottleneck link. With TurboPlay disabled, the streaming rate (see Figure 6(a)) of the 512 kbps media during the buffering period is approximately 600 kbps. This higher rate is maintained for about 72 seconds before reverting to a lower rate for the rest of the streaming session. With TurboPlay enabled, the streaming rate during the buffering period approaches 1500 kbps, utilizing the full network capacity. This higher rate lasts for about 8 seconds before reverting to the steady streaming rate for the rest of the session. Clearly, the TurboPlay mechanism makes the buffering behaviour more aggressive. Consequently, it reduces the buffering startup delay.

One side effect of the faster buffering rate is the risk of packet loss. Figure 6(b) shows the cumulative packet loss statistics from the foregoing experiment. With TurboPlay disabled, there were no packet losses observed. With TurboPlay enabled, there were about 60 packet losses observed. All the packet losses occurred during the startup period, where the buffering rate saturated the network capacity.

B. *Experiments with TCP Cross Traffic*

This section studies TCP friendliness by examining the interaction between UDP RealVideo streams and TCP cross traffic. The TCP variant used in these experiments is TCP SACK. Unless stated otherwise, the RealVideo results reported here are with the TurboPlay option enabled (as this option is enabled, by default, for all broadband clients).

In these experiments, we study the interaction between one RealVideo stream and a single bulk-transfer TCP flow. Our main observation is that RealVideo streams are not TCP friendly; they consume a disproportionate share of the available network bandwidth.

To establish TCP steady state, the TCP traffic was started approximately 10 seconds before the media stream. In the time series graphs, time zero represents the time at which the RealPlayer first requests data from the server. Negative time values represent time before this event.

Figure 7 illustrates the TCP unfriendliness of the RealVideo streams. This experiment uses the 1000 kbps media stream on a 1500 kbps network link. Observe that before the media

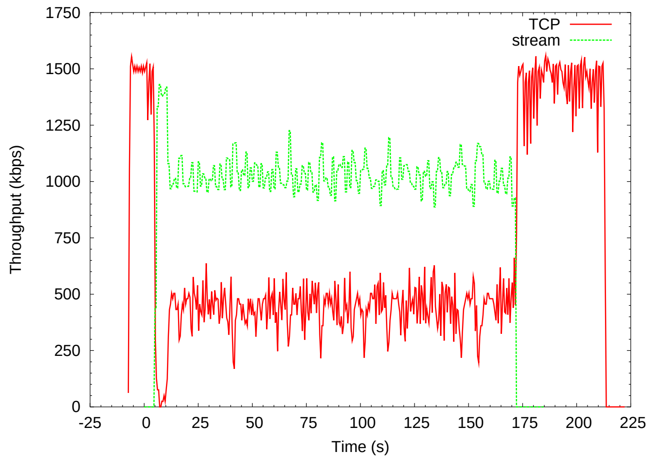


Fig. 7. Bandwidth competition between a 1000 kbps RealVideo stream and TCP flow over a 1500 kbps bottleneck link

stream begins, the TCP traffic occupies the full link bandwidth, attaining a peak throughput of 1500 kbps. After the streaming starts, the RealVideo stream consumes about 1000 kbps, leaving about 500 kbps of bandwidth for the TCP flow.

We also studied the influence of the TurboPlay option on the TCP friendliness of the RealVideo streams. For illustration, we consider the 512 kbps media stream and the 1500 kbps bottleneck link. Figure 8 shows sample throughput versus time plots from our experiments. Figure 8(a) shows results with TurboPlay enabled. Figure 8(b) shows results with TurboPlay disabled. Figure 8(c) shows the cumulative packet loss results both with and without the TurboPlay feature.

When TurboPlay is enabled (Figure 8(a)), there is a higher streaming rate during the initial buffering period. The TCP flow suffers dramatically during this startup period. When the media stream returns to its encoded bit rate, the TCP flow soon recovers, consuming the remaining network bandwidth (1000 kbps).

When the TurboPlay option is disabled, very different behaviour is observed in several experiments. In particular, the TCP cross traffic completely overruns the media stream, as illustrated in Figure 8(b). The media player in this case reverts to audio-only mode (note the periodic bandwidth spikes for the stream).

These results show that the TurboPlay feature fundamentally changes the behaviour of RealVideo streams. Based on our experiments, we conjecture that when TurboPlay is disabled, the media client uses packet losses as an indication of congestion, and reduces bandwidth consumption accordingly (and somewhat aggressively). Thus, the TCP cross traffic can potentially monopolize the bottleneck link, as illustrated in Figure 8(b). In many experiment runs, we observed that the RealVideo stream reverts to audio-only mode (with no video playback) and does not recover from this mode even when the TCP flow ends.

With TurboPlay disabled, the streams can become overly friendly to TCP, and essentially become non-functional. Therefore, it is no surprise that TurboPlay is enabled in the default configuration.

Enabling TurboPlay apparently suppresses the application-layer congestion control mechanism implemented in the RealPlayer client. While this configuration provides good quality media playback, it is not TCP friendly. From these and many other experiments, we conclude that single-bit-rate RealVideo streams are not TCP friendly.

An alternative to single-bit-rate media streams is SureStream technology. A SureStream media object contains many other media objects. The client and server can dynamically determine the appropriate stream encoding rate based on the client's available bandwidth. Our final experiment studies the TCP friendliness of SureStream streams.

Figure 9(a) and (b) illustrates the achieved throughput for the RealVideo SureStream stream with TurboPlay enabled and disabled, respectively. Figure 9(c) shows the cumulative packet loss results versus time for these two cases.

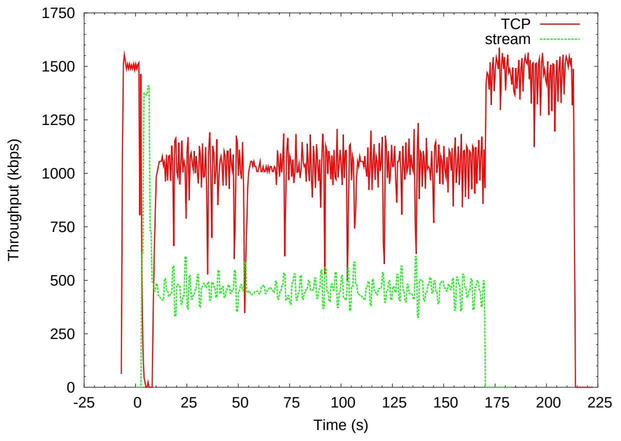
Figure 9 shows that the SureStream stream is not TCP friendly. When TurboPlay is enabled, the SureStream media is able to effectively "shut down" the competing TCP flow, in a manner similar to that of a single-bit-rate stream encoded at 1500 kbps. However, using SureStream with TurboPlay disabled does provide some compromise. In this setting, the SureStream video initially starts at 1500 kbps, but soon drops to 1200 kbps, and finally settles down to periodically oscillate between 750 kbps and 1100 kbps. When the media stream ends, the TCP flow reclaims the full network bandwidth. While not perfectly fair, the SureStream behaviour with TurboPlay disabled is certainly "friendlier" to TCP than a single-bit-rate stream.

V. Conclusions

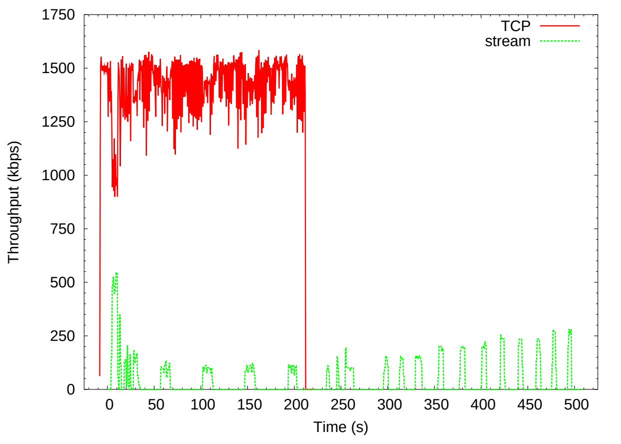
This paper presented a high-level characterization of RealVideo streams using an emulated network. The central bottleneck link, the media encoding rate, and the TurboPlay option were the variables examined.

The main observations from this study are as follows:

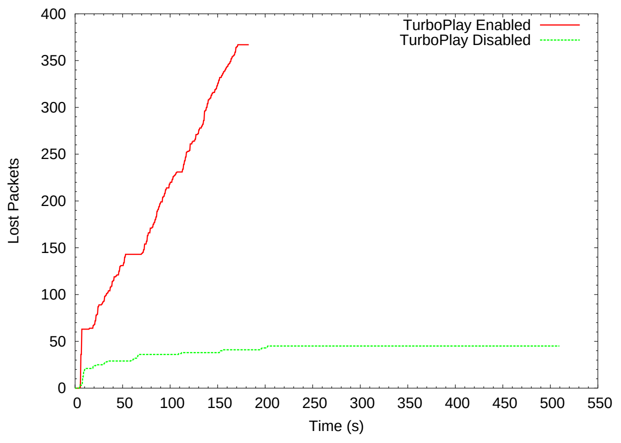
- The buffering behaviour of the client player varies depending on the encoding rate of the media stream. For lower-bandwidth streams the buffering rate is approximately 2-4 times the encoded rate, to a maximum of approximately 600 kbps. For higher-bandwidth streams, the buffering takes place at the encoding rate.
- The TurboPlay option in the client player changes the buffering behaviour. Regardless of stream encoding rate, the buffering bit rate increases to the maximum rate that the bottleneck allows.
- If the media encoding rate exceeds the bottleneck capacity, the client behaviour observed depends on the media bit rate. Lower-rate media streams resort to audio-only playback, while higher-rate media streams continue to play the video stream, but at a lower quality.
- With the TurboPlay option enabled, RealVideo streams are not TCP friendly. The UDP media flows take as much bandwidth as they need, to the detriment of TCP.



(a) TurboPlay enabled

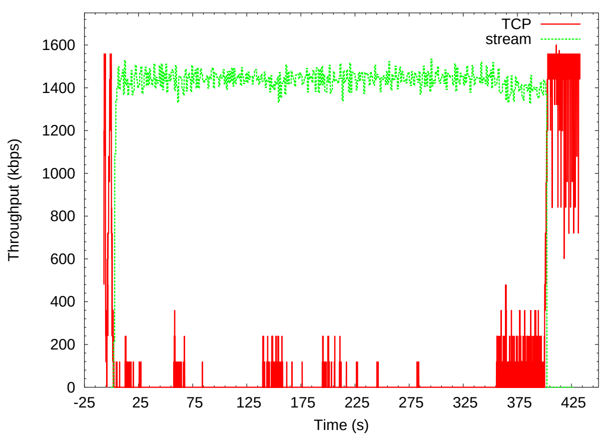


(b) TurboPlay disabled

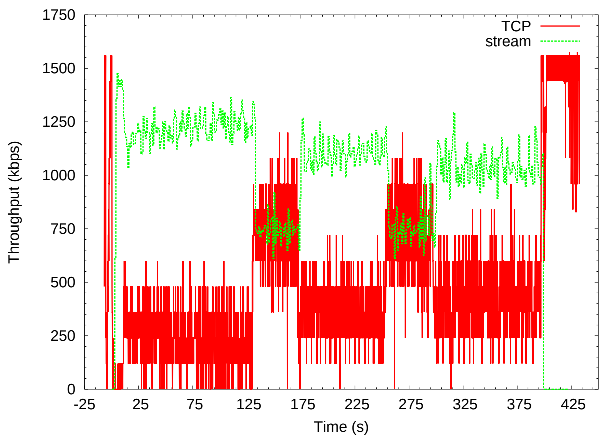


(c) Cumulative packet loss

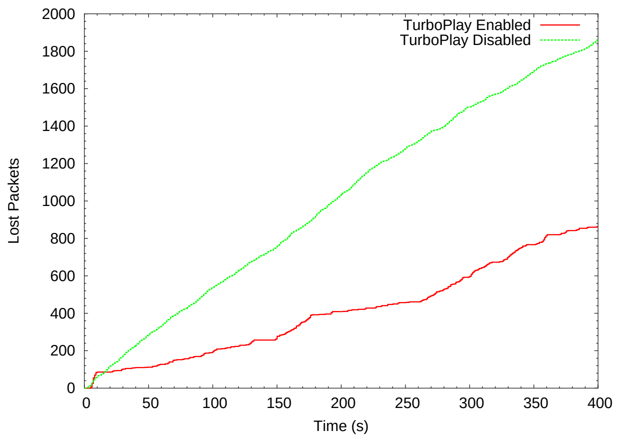
Fig. 8. Effect of TurboPlay option on TCP friendliness of RealVideo streams (512 kbps stream/1500 kbps bottleneck link)



(a) TurboPlay enabled



(b) TurboPlay disabled



(c) Cumulative packet loss

Fig. 9. Effect of TurboPlay option on TCP friendliness of SureStream (1500 kbps bottleneck link)

- With the TurboPlay option disabled, RealVideo streams can become TCP friendly; however, the playback performance may degrade substantially.
- Real's SureStream – an application-level feature to adjust the stream rate during playback – provides some compromises, reducing the bit rate to allow some TCP throughput, but is not strictly TCP friendly.

Ongoing work is expanding our experimental study to consider other commercial media streaming products, and more realistic network configurations. We are also investigating the use of media streaming proxies to provide end-to-end TCP friendliness on an application-independent basis.

ACKNOWLEDGEMENTS

Financial support for this research was provided by the University of Calgary, the Informatics Circle of Research Excellence (iCORE) in the Province of Alberta, the Natural Sciences and Engineering Research Council (NSERC) of Canada, and the Canada Foundation for Innovation (CFI).

The authors thank the anonymous reviewers for their suggestions and feedback which helped improve this paper. The authors also thank Nayden Markatchev and Gwen Houtzager for technical support related to this work.

REFERENCES

[1] R. Agarwal, B. Hefta-Gaub, and D. Stammen. RealMedia File Format. IETF Draft draft-heftagaub-rmff-00.txt, March 1998.

[2] B. Braden, D. Clark, J. Crowcroft, B. Davis, S. Deering, D. Estrin, S. Floyd, V. Jacobson, G. Minshall, C. Partridge, L. Peterson, K. Ramakrishnan, S. Shenker, J. Wroclawski, and L. Zhang. Recommmendations on Queue Management and Congestion Avoidance in the Internet. RFC 2309, April 1998.

[3] M. Chesire, A. Wolman, G. Voelker, and H. Levy. Measurement and Analysis of a Streaming Media Workload. In *Proc. of USENIX USITS*, San Francisco, USA, March 2001.

[4] J. Chung, M. Claypool, and Y. Zhu. Measurement of the Congestion Responsiveness of RealPlayer Streaming Video Over UDP. In *Proc. of International Packet Video Workshop*, Nantes, France, April 2003.

[5] S. Floyd and K. Fall. Promoting the Use of End-to-End Congestion Control in the Internet. *IEEE/ACM Trans. on Networking*, 7(4):458–472, 1999.

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based Congestion Control for Unicast Applications. In *Proc. of ACM SIGCOMM Conference*, Stockholm, Sweden, August 2000.

[7] T. Kuang and C. Williamson. Hierarchical Analysis of RealMedia streaming traffic on an IEEE 802.11b wireless LAN. *Computer Communications*, 27(6):538–548, April 2004.

[8] M. Li, M. Claypool, and R. Kinicki. MediaPlayer versus RealPlayer – A Comparison of Network Turbulence. In *Proc. of ACM Internet Measurement Workshop*, Marseille, France, November 2002.

[9] M. Li, M. Claypool, R. Kinicki, and J. Nichols. Characteristics of Streaming Media Stored on the Web. *ACM Transactions on Internet Technology (to appear)*, 2005.

[10] J. Nichols, M. Claypool, R. Kinicki, and M. Li. Measurements of the congestion responsiveness of windows streaming media. In *Proc. of ACM NOSSDAV*, pages 94–99, County Cork, Ireland, June 2004.

[11] RealNetworks. RealSystem production guide: RealSystem release 8 with w ith RealProducer 8.5. http://service.real.com/help/library/index.html.

[12] RealNetworks. RealVideo 10: Technical Overview. http://docs.real.com/docs/rn/rv10/RV10%5FTech%5FOverview.pdf, 2003.

[13] R. Rejaie, M. Handley, and D. Estrin. RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet. In *Proc. of IEEE Infocom*, New York, USA, March 1999.

[14] R. Simmonds, R. Bradford, and B. Unger. Applying Parallel Discrete Event Simulation to Network Emulation. In *Proc. of ACM/IEEE/SCS PADS*, pages 15–22, Bologna, Italy, May 2000.

[15] K. Sripanidkulchai, B. Maggs, and H. Zhang. An Analysis of Live streaming Workloads on the Internet. In *Proc. of ACM IMC*, pages 41–54, Taormina, Italy, November 2004.

[16] E. Veloso, V. Almeida, W. Miera, A. Bestavros, and S. Jin. A Hierarchical Characterization of a Live Streaming Media Workload. In *Proc. of ACM Internet Measurement Workshop*, Marseille, France, November 2002.

[17] Y. Wang, M. Claypool, and Z. Zuo. An Empirical Study of RealVideo Performance Across the Internet. In *Proc. of ACM Internet Measurement Workshop*, pages 295–309, San Francisco, USA, November 2001.

ABOUT THE AUTHORS

Sean Boyden is currently an M.Sc. student at the University of Calgary. He received his B.Sc. in Computer Science from the University of Northern British Columbia in Prince George, BC, Canada in 2004. His research interests include network protocols, wireless sensor networks, concurrency, and system modeling.

Anirban Mahanti received the B.E. degree in Computer Science & Engineering from the Birla Institute of Technology, Ranchi, India in 1997, and the M.Sc. and Ph.D. degrees in Computer Science from the University of Saskatchewan, Saskatoon, Canada, in 1999 and 2004, respectively. He is an Assistant Professor in the Department of Computer Science at the University of Calgary, Calgary, Canada. His research interests are in the areas of performance evaluation of distributed computer systems and computer networks. His specific research interests include multimedia streaming systems, Web performance, network measurement, network modeling, network architectures, network protocols, and distributed systems.

Carey Williamson is an iCORE Professor in the Department of Computer Science at the University of Calgary, specializing in Broadband Wireless Networks, Protocols, Applications, and Performance. He holds a B.Sc.(Honours) in Computer Science from the University of Saskatchewan, and a Ph.D. in Computer Science from Stanford University. His research interests include Internet protocols, wireless networks, network traffic measurement, network simulation, and Web server performance.