MEASUREMENT OF THE CONGESTION RESPONSIVENESS OF REALPLAYER STREAMING VIDEO OVER UDP

Jae Chung, Mark Claypool and Yali Zhu

Computer Science Department
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609, USA
{goos|claypool|yaliz}@cs.wpi.edu

ABSTRACT

The growth in power and connectivity of today's PCs promises a continued increase in the growth of streaming media over the Internet. Hand-in-hand with the increase in streaming media comes the impending threat of unresponsive UDP traffic, often cited as the major threat to the stability of the Internet. The responsiveness of commercial streaming media applications, such as RealNetworks' RealPlayer, will play an important role in the network impact of streaming media. Unfortunately, there are few empirical studies that analyze the responsiveness, or lack of it, of commercial streaming media. In this work, we measure the responsiveness of RealVideo over UDP by measuring the performance of numerous streaming video clips selected from a variety of RealServers on the Internet. By varying the bottleneck bandwidth to the player, we are able to analyze the *TCP-Friendliness* of RealVideo over UDP and correlate the results with network and application layer statistics. We find that most streaming RealVideo clips are not bandwidth constrained for typical broadband connections. In times of congestion, most RealVideo UDP streams respond to Internet congestion by reducing the application layer encoding rate, and streams with a minimum encoding rate less than the fair bandwidth share usually achieve a TCP-Friendly rate. In addition, our analysis suggests that a reason streaming applications choose not to use TCP is that the TCP API hides network information, such as loss rate and round-trip time, making it difficult to estimate the available bandwidth for effective media scaling.

1. INTRODUCTION

The growth in power and connectivity of today's computers has enabled streaming video across the Internet to the desktop. Increasingly, users can access online video clips through a Web browser by simply clicking on a link and having the Web browser start up an associated video player. Web sites today offer streaming videos of news broadcasts, music television, live sporting events and more. For example, in 2001 an estimated of 350,000 hours of online entertainment was broadcast each week over the Internet [17], with countless more hours downloaded on-demand.

While voice quality audio typically operates over a narrow range of bandwidths (32-64 Kbps), video operates over a much wider range of bandwidths. Video conferences and Internet videos stream at about 0.1 Mbps[1], VCR quality videos at about 1.2 Mbps[2], broadcast quality videos at about 2-4 Mbps[3], studio quality videos at about 3-6 Mbps[3], and HDTV quality videos at about 25-34 Mbps[3]. Uncompressed video can require hundreds and even thousands of Mbps. Thus, video applications have the potential to demand enormous amounts of bandwidths, often greater than the available network capacity, but also have the potential to reduce their data rates when bandwidth is constrained.

While TCP is the de facto standard transport protocol for typical Internet applications, there are as of yet no widely accepted rate-based transport protocols for streaming media applications. Unlike typical Internet traffic, streaming video is sensitive to delay and jitter, but can tolerate some data loss. In addition, streaming video typically prefers a steady data rate rather than the bursty data rate often associated with window-based network protocols. Recent research has proposed rate-based TCP-Friendly protocols in the hope that streaming media applications will use them [19, 9], but such protocols are not yet widely part of any operating system distribution. For these reasons, streaming video applications often use UDP as a transport protocol rather than TCP. Moreover, with the use of repair techniques [2, 12, 16], packet losses can be partially or fully concealed, reducing the impact of loss on the quality of the video by the user, and thus reducing the incentive for multimedia ap-

[1] H.261 and MPEG-4
[2] MPEG-1
[3] MPEG-2

plications to lower their bandwidth in the presence of packet loss during congestion.

Potentially high-bandwidth video over UDP using repair techniques suggests that video flows may not be *TCP-friendly* or, even worse, that video flows may be unresponsive to network congestion. In the absence of end-to-end congestion control, TCP flows competing with UDP flows reduce their sending rates in response to congestion, leaving the unresponsive UDP flows to expand to use the vacant bandwidth, or, worse, contribute to congestion collapse of the Internet [8].

In light of this, recent research has explored router queue management approaches to identify and police unresponsive flows [13, 20, 7, 15, 3]. Such research often models unresponsive flows as transmitting data at a constant packet size and constant bit rate (CBR) or, as "firehose" applications, transmitting at an unyielding, maximum rate. However, commercial media products have been shown to not be strictly CBR [14, 11], and, although using UDP, may respond to congestion at the application layer. A better understanding of the traffic rates and responsiveness of current streaming media applications may help create more effective network techniques to handle unresponsive traffic.

The responsiveness of commercial streaming media products will play an important role in the impact of streaming media on the Internet. The use of commercial streaming products, such as the Microsoft Windows Media Player and RealNetworks RealPlayer, has increased dramatically [10]. Communication with commercial streaming media product developers has been ineffective in providing adequate, scientific information on the congestion responsiveness, leaving measurement as the next viable option. While there have been some studies characterizing streaming traffic [4, 14, 11, 23], there are few empirical studies that analyze the responsiveness, or lack of it, of current streaming media products.

This study evaluates the responsiveness of RealVideo streaming over UDP by comparing it to the data rates of TCP under the same network conditions. We set up a network testbed where two clients, one using UDP and the other using TCP, streamed video through a network router we control, connected to the Internet via a broadband connection. We varied the bottleneck bandwidth to the clients by limiting the bandwidth of the router's outgoing connection, allowing us to explore a range of congestion situations. The two clients then simultaneously streamed hundreds of videos selected with a variety of content and encoding formats from a diverse set of Web servers, while measuring packet loss rates and round-trip times as well as application level statistics such as encoded bandwidths and frame rates. By using the TCP stream as the desired level of responsiveness, we are able to quantify the responsiveness of the video stream over UDP and correlate the results with network and application statistics.

In analyzing our data, we make several contributions to better understanding the characteristics of potentially unresponsive streaming video on the Internet. We find that overall, most streaming RealVideo clips are not constrained by bandwidth from a typical broadband connection, resulting in a fair share of link bandwidth for both RealVideo over UDP and TCP. In times of congestion, most streaming RealVideo over UDP does respond to Internet congestion by reducing the application layer encoding rate. We also find several key incentives for video streams to use UDP rather than TCP, suggesting that potentially unresponsive streaming media over UDP will likely persist for some time.

The rest of this paper is organized as follows: Section 2 presents background on RealPlayer needed to understanding our results; Section 3 describes our approach to obtain a wide-range of Internet measurements; Sections 4 and 5 present and analyze the measurement data obtained; Section 6 discusses our findings; Section 7 summarizes our conclusions and Section 8 presents possible future work.

2. REALVIDEO BACKGROUND

RealPlayer, provided by RealNetworks[4], is the most popular streaming media player on the US Internet, with over 47% of the commercial market share [10]. RealVideo content providers create streaming videos using a variety of possible video codecs, convert it to RealNetworks' proprietary format and place it on an Internet host running RealServer. During creation, content providers select target bandwidths appropriate for their target audience and specify other encoding parameters, such as frame size and frame rate, appropriate for their content. A RealServer then streams the video to a user's RealPlayer client upon request.

RealServer and players primarily use Real Time Streaming Protocol[5] (RTSP) for the session layer protocol. Occasionally, RealServer will use HTTP for metafiles or HTML pages, and HTTP may also be used to deliver clips to RealPlayers that are located behind firewalls. For this measurement study, all the video clips selected used RTSP, as described in Section 3.1.

At the transport layer, RealServer uses both TCP and UDP for sending data. The initial connection is often in TCP, with control information then being sent along a two-way TCP connection. The video data itself is sent using either TCP or UDP. By default, the actual choice of transport protocol used is determined automatically by the RealPlayer and RealServer, resulting in UDP about 1/2 the time and TCP the other half [23]. The decision making process RealPlayer makes in choosing either UDP or TCP is not known, and may be interesting future work. The choice

[4]http://www.real.com/

[5]http://www.rtsp.org/

of UDP or TCP can also be manually specified by the user[6]. For our study, we specifically set RealPlayer to use UDP in some cases and TCP in others, as described in Section 3.2.

RealSystem supports an application level media scaling technology called *SureStream* in which a RealVideo clip is encoded for multiple bandwidths [6, 18]. When streaming a SureStream RealVideo clip, RealServer determines which encoded stream to use based on feedback from RealPlayer regarding the client end-host network. The actual video stream served can be varied in mid-playout, with the server switching to a lower bandwidth stream during network congestion and then back to a higher bandwidth stream when congestion clears. We study the flexibility of SureStream scaling in Section 5.3.

For each video clip, RealPlayer keeps a buffer to smooth out the video stream because of changes in bandwidth, lost packets or jitter. Data enters the buffer as it streams to RealPlayer, and leaves the buffer as RealPlayer plays the video clip. If network congestion reduces bandwidth for a few seconds, for example, RealPlayer can keep the clip playing with the buffered data. If the buffer empties completely, RealPlayer halts the clip playback for up to 20 seconds while the buffer is filled again. We measure the rate at which RealPlayer fills the buffer in Section 5.4.

3. APPROACH

In order to empirically measure the responsiveness of RealVideo over UDP, we employed the following methodology:

- Select numerous RealVideo URLs that use the Real Time Streaming Protocol (RTSP) using well-known Web search engines (see Section 3.1).
- Construct an environment for measuring the responsiveness of RealVideo over UDP by comparing it to TCP under the same network conditions (see Section 3.2).
- Construct a "media scaling" environment for comparing the application layer behavior of non-competing RealVideo over UDP or TCP (see Section 3.3).
- Iteratively play the selected RealVideo clips in both environments with different bottleneck bandwidths and analyze the results (see Section 4 and Section 5).

3.1. RealVideo Clip Playlist

We desired a relatively realistic environment in which we could measure and compare the network layer responsiveness of RealVideo over UDP with that of long-lived TCP flows sharing the same network path. In a stand-alone environment in which we could precisely control the network conditions from the server to the client, the encoded content and server platform chosen might impact performance more than the network, resulting in inaccurate conclusions about the Internet at large. Thus, we decided to use publicly available Internet RealVideo servers and clips as the traffic sources.

To form a clip playlist, we searched for RealVideo clips (URLs) accessible through Web pages using well-known search engines, such as Yahoo and Google, and randomly selected 100 RTSP RealVideo URLs from the search results. Of the selected URLs, 76 were from the United States, 9 from Canada, 8 from the United Kingdom, 6 from Italy, and 1 from Germany. While our selection method of using US/English based commercial search engines likely influenced the predominance of North American URLs, our RealPlayer clients ran from North America, and it is likely that there is typically similarly strong locality of access for most streaming players.

For the clips selected, the median clip length was about 3 minutes, while the shortest and longest clips played out in 20 seconds and 30 minutes, respectively. Other statistics on the selected RealVideo clips are available in Section 4, Section 5.3 and [5].

3.2. Responsiveness of RealVideo over UDP Measurement Environment

Ideally, we sought an environment in which to measure the network layer responsiveness of RealVideo over UDP by comparing it to that of long-lived TCP flows under the same network conditions. Since Internet network conditions are volatile, we wanted to run simultaneous RealVideo over UDP and bulk TCP flows along the same network path, rather than run consecutive UDP and TCP flows. Unfortunately, public RealServers do not typically support bulk TCP transfers making it difficult to ensure a bulk TCP would use the same path as a RealPlayer UDP. Instead, we used RealVideo over TCP as the yardstick with which to compare RealVideo over UDP. Since RealVideo applications are rate-based, at the network level RealVideo over TCP may request the same as or less bandwidth than a bulk TCP transfer under the same network conditions, providing a "lower bound" on the bandwidth a bulk TCP transfer would use.

We had two RealPlayers, one using UDP and the other using TCP, simultaneously stream a video clip from the same RealServer along the same network path, while we captured network and application statistics. As depicted in Figure 1, the two RealPlayers ran on separate PCs attached to the same 10 Mbps hub. Both PCs were equipped with a Pentium III 700 MHz processor, 128 MB RAM and a UDMA-66 15 GB hard disk, and were running Linux kernel version 2.4. Both PCs ran RealPlayer version 8.0.3, with one Re-

[6] http://service.real.com/help/player/plus_manual.8/rppmanual.htm

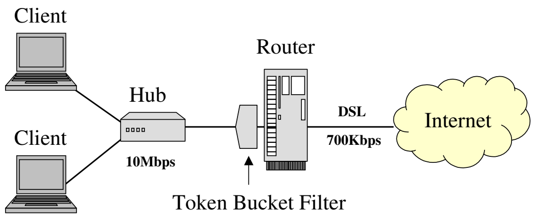

Fig. 1. Testbed Network Setup: Environment to Measure the Responsiveness of RealVideo

alPlayer configured to use UDP and the other RealPlayer configured to use TCP.

The hub facilitated capturing network layer performance since packets destined to either PC were broadcasted to both PCs. We ran `tcpdump`[7], a well-known network packet sniffer, on one PC to filter and log the video stream packets. As the RealVideo packet format is proprietary, we were unable to obtain sequence numbers and, hence, loss information, from the packet traces themselves. We did run `tcptrace` on the tcpdump data, but it only provides statistics on the very sparse amount of RTSP control traffic from the client to the server and not statistics on the data stream itself. Instead, during each clip, we ran a `ping` at 1 second intervals to the server to obtain samples of the round-trip time (RTT) and packet loss rate. During pilot studies, we confirmed that the RTTs and loss rates obtained via `ping` samples were comparable to those obtained via `tcptrace`. Also, we verified that the packet filtering and logging did not induce much CPU or disk load and did not interfere with the video playout. At the end of each RealVideo stream, information such as the IP packet size and arrival time were extracted from the `tcptrace` log using `ethereal`[8] and processed to obtain network layer statistics, such as throughput.

In order to control network congestion, we considered adding background traffic along the path from the client to the servers. However, as discussed, most RealServers do not simultaneously provide other file services making it difficult to add congestion-causing traffic to servers in a controlled manner. Instead, to consistently control the incoming bandwidth, we set up a private Linux router connected to a commercial DSL 700 Kbps network to enable us to create constrained bandwidth situations. The router was configured to use network address translation (NAT) to eliminate the possibility that packets from the competing TCP and UDP streams to be routed differently. We attached a software implementation of a Token Bucket Filter (TBF)[9] to the Ethernet card at the internal network of the router. The TBF queue size was set to 10 Kbytes and the *burst allowed* (the maximum number of tokens available during idle times) was set to 1600 Bytes, slightly larger than a typical 1500 Byte MTU. The *token rate* (available bandwidth) was set to 600 Kbps, 300 Kbps, 150 Kbps and 75 Kbps. Note, since we have two streaming flows, one TCP and one UDP, competing, their fair bandwidth share is approximately half of each bottleneck bandwidth.

For each DSL-TBF configuration, we carried out two sets of measurements, where each set played all video clips in the playlist.

3.3. Media Scaling Measurement Environment

Streaming video can adjust to the available bandwidth during congestion by *media scaling* where video encoding is switched to a lower rate. As mentioned in Section 2, RealSystems uses a media scaling technology called *SureStream* in which a RealVideo clip is encoded for multiple bandwidths [18]. The actual video stream served can be varied in mid-playout, with the server switching to a lower bandwidth stream during network congestion and then back to a higher bandwidth stream when congestion clears.

To study media scaling in RealPlayer we used *RealTracer*[10], developed for a previous study [23], which plays RealVideo streams and records application level statistics, including encoding rate. One of the client machines was booted with Microsoft Windows ME and equipped with RealPlayer 8 Basic version 6.0.9 and RealTracer version 1.0. We then ran a non-competing, single UDP or TCP stream for each URL in the playlist, while limiting the TBF incoming bandwidth to 35 Kbps[11], since the highest encoded bandwidth for all clips that did media scaling was above 35 Kbps. We tried other TBF rates such as 25 Kbps, 150 Kbps and 300 Kbps to verify we measured all possible scale levels (or encoded bandwidths) used for clip playouts. However, only 2 sets of measurements, TCP for the entire playlist and UDP for the entire playlist, on the 35 Kbps DSL-TBF configuration were used to characterize the responsiveness of RealVideo media scaling (see Section 5.3).

4. RESULTS

Over the course of 2 months, we streamed over a total of 200 hours of video from a cumulative total of over 4000 video clips. Of the original set of 100 video clips, 1 clip could not be served using UDP, perhaps because of server firewall restrictions. Also, 20 other clips became completely unavailable sometime after the initial selection before the

[7] http://www.tcpdump.org/
[8] http://www.ethereal.com/
[9] http://www.linuxpowered.com/archive/howto/Adv-Routing-HOWTO-7.html#ss7.3
[10] http://perform.wpi.edu/real-tracer/
[11] The queue was set to 5 Kbytes for the 35 Kbps DSL-TBF configuration.

experiments were complete. We removed these clips from further analysis.

Of the remaining 79 clips in the playlist, about 30% of their servers did not respond to `ping` packets, making them unavailable for loss and round trip time (RTT) analysis. For all RTT and loss analysis in this report, we removed the data from these clips. However, we did use the other data recorded on these clips for other analysis.

Comparing the average RTTs obtained via `ping` probes for each bottleneck bandwidth, the 75 Kbps connection had the highest round-trip times. The median RTTs for the 75, 150, 300 and 600 Kbps configurations were 450, 340, 130 and 100 ms respectively. For the 150-600 Kbps configurations, about 33% of the clips had the same RTT regardless of the bottleneck bandwidth since these clips stream at less than 150 Kbps, and therefore do not suffer additional queuing delays at the router. For the remaining 67% of the clips, the lower the bottleneck bandwidth the higher the queuing delays, caused primarily by the 10 Kbyte buffer at the bottleneck router.

Summarizing the loss rates obtained via `ping` probes for each bottleneck bandwidth, the median loss rate for any configuration was less than 2%. About 37% of the clips played with low bottleneck bandwidths had no loss, while about 50% of the clips played at higher bottleneck bandwidths had no loss. Overall loss rates increased about 1% for each decrease in bottleneck bandwidth as bandwidths decreased from 600 Kbps to 300 Kbps to 150 Kbps to 75 Kbps. The low loss rates, even at low bandwidth connections, implies that most of the RealVideo UDP streams adapted to the available bandwidth, and is investigated in depth in Section 5. Due to space constraints, we do not present further details on the RTT and loss results here, but refer the interested reader to [5].

5. ANALYSIS

In analyzing the responsiveness of RealVideo over UDP, we first analyze bandwidth aggregated over all clips and then for individual clip pairs (Section 5.1). We then analyze the TCP-Friendliness of RealVideo over UDP (Section 5.2). Moving to the application layer, we analyze the application scaling behavior (Section 5.3). Lastly, we measure the initial buffering rate compared with the steady playout rate (Section 5.4).

5.1. Bandwidth

Figure 2 depicts Cumulative Density Functions (CDFs) of the per-clip average bandwidth used by TCP and UDP for bottleneck bandwidths of 600, 300, 150 and 75 Kbps. The TCP and UDP distributions are nearly the same for the 600 Kbps bottleneck bandwidths. However, as bandwidth becomes more constrained, the distributions separate, with UDP having a consistently higher distribution of bandwidths than TCP.

We next analyze the head-to-head bandwidth for each pair of (TCP, UDP) clips. For each clip pair, in Figure 3 we plot an (x,y) point where x is the average bandwidth used by the TCP stream and y is the average bandwidth used by the UDP stream. The points for each bottleneck bandwidth are depicted by a different point style. The dashed 45 degree line provides a reference for bandwidth equally shared by TCP and UDP. Points above the line (top left of the graph) indicate UDP received more average bandwidth while points below the line (bottom right of the graph) indicate TCP received more average bandwidth. The distance from the line indicates the magnitude of the average bandwidth difference.

From Figure 3, while there are some points that lie along the equal bandwidth line, there are many cases of bandwidth disparity. The highest bandwidth playouts for the 600 Kbps bottleneck bandwidths had the greatest bandwidth disparities. For the 600 Kbps bottleneck bandwidths, there are visually as many points below the equal bandwidth line where TCP received more bandwidth as there are above the equal bandwidth line where UDP received more bandwidth. For the lower bottleneck bandwidths, there are visually considerably more points above the equal bandwidth line, indicating UDP received more bandwidth.

We next analyze the bandwidth disparity relative to the bottleneck bandwidth available. For each clip pair, we subtract the UDP average bandwidth from the TCP average bandwidth and divide the difference by the bottleneck bandwidth. Thus, equal sharing of bandwidth has a value of zero, a value of -1 indicates UDP got the entire bottleneck bandwidth, and a value of +1 indicates TCP got the entire bottleneck bandwidth. Figure 4 depicts CDFs of the normalized bandwidth differences for each bottleneck bandwidth.

For the 600 Kbps bottleneck bandwidth, about 40% of the clips shared the bandwidth equally. As indicated by the region in the top right, about 30% of TCP got more bandwidth than their counterpart UDP clips while about 20% of the UDP clips got more bandwidth than their counterpart TCP clips, as indicated by the region in the bottom left. The greatest bandwidth disparity was approximately half the bottleneck bandwidth.

For the lower bottleneck bandwidths, there were increasingly fewer clips with equal bandwidth. The UDP clips got substantially more bandwidth than did their TCP counterparts, as indicated by the large areas under the distributions on the bottom left. For the 300 Kbps bottleneck bandwidth, about 60% of the UDP clips got more bandwidth than their TCP counterparts, and for the 150 Kbps and 75 Kbps bottleneck bandwidths, about 70% of the UDP clips got more bandwidth than their TCP counterparts. For the

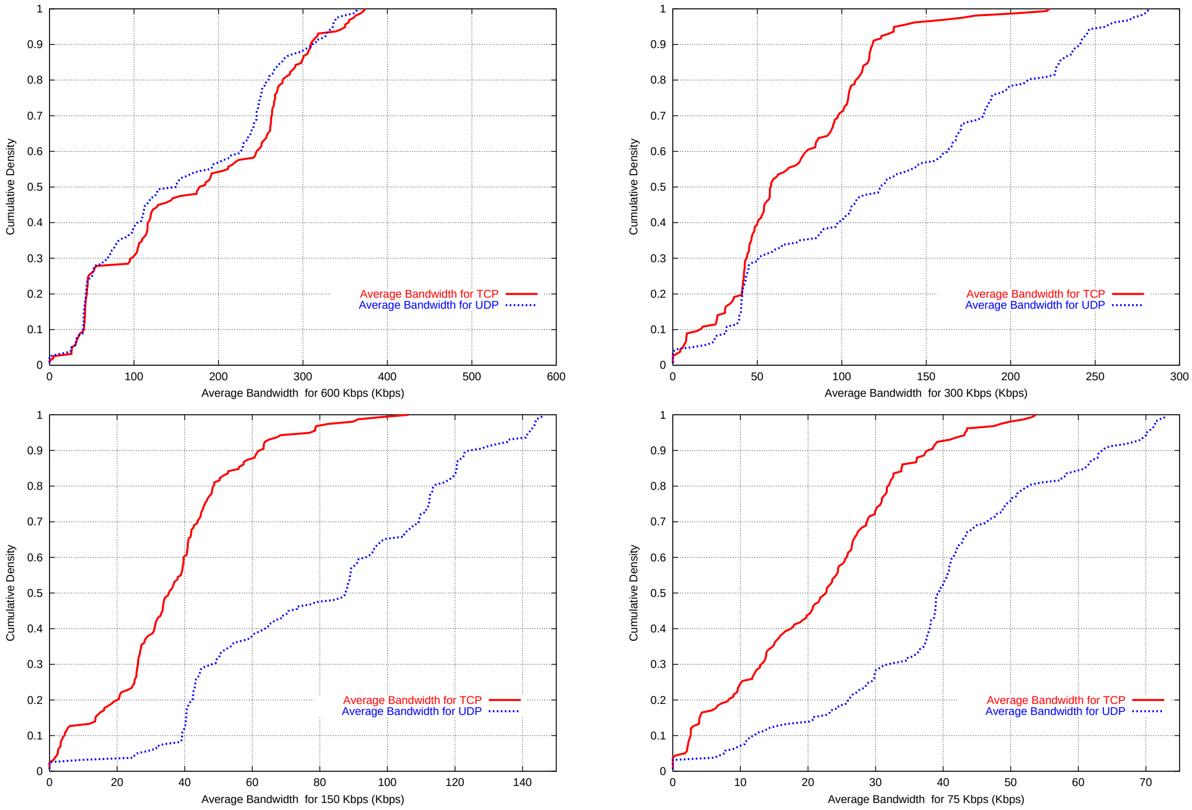

Fig. 2. CDFs of Average Bandwidth for Bottleneck Bandwidths of 600, 300, 150, and 75 Kbps

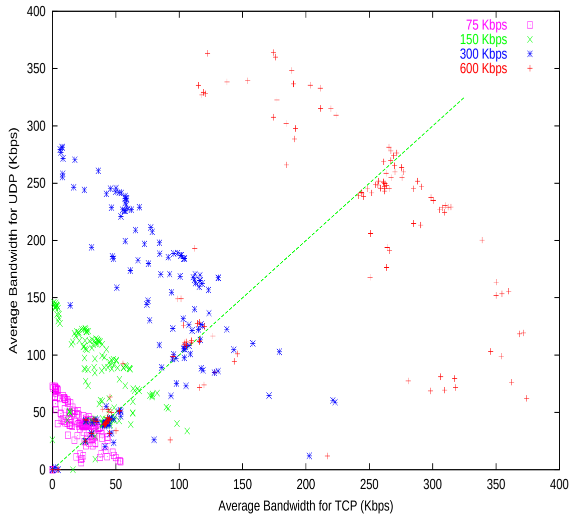

Fig. 3. Head-to-Head Average Bandwidth (All Runs)

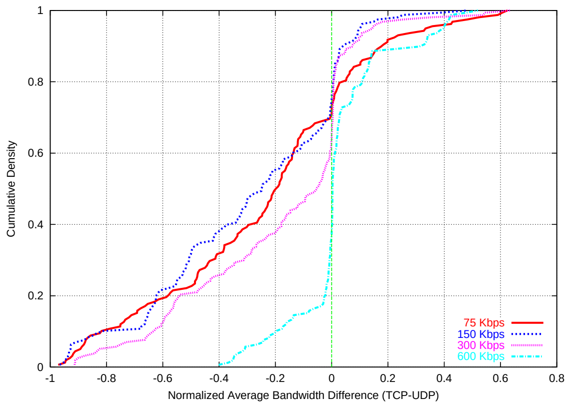

Fig. 4. CDF of the Difference (TCP - UDP) in the Average Bandwidth, Normalized by the Bottleneck Bandwidth (All Runs)

300, 150 and 75 Kbps bottleneck bandwidths, about 20% of the UDP clips got twice the normalized bandwidth of their TCP counterparts. For the 150 and 75 Kbps bottleneck bandwidths, about 20% of the UDP clips received over 80% more of the normalized bandwidth than their TCP counterparts. However, even for the lowest bottleneck bandwidths, there were still cases where TCP got more bandwidth than their UDP counterparts, as depicted by the areas above the distributions in the upper right.

In general, as bandwidth becomes constrained, streaming RealVideo clips over UDP receive relatively more bandwidth than do streaming streaming RealVideo clips over TCP. However, further limiting bandwidth does not significantly change the UDP vs. TCP bandwidth allocation ratio. A significantly large number of the UDP video streams are able to adapt to reduced bandwidths without causing increased congestion. Moreover, in all cases, streaming RealVideo over UDP sometimes receives less bandwidth than do competing TCP flows, especially for higher bottleneck bandwidths.

Our analysis of round-trip times and loss rates obtained by the `ping` samples show modest correlations for both round-trip times and bandwidth disparity and loss rates and bandwidth disparity. In other words, as round-trip times and loss rates increase, streaming RealVideo clips over UDP receive relatively more bandwidth than do streaming RealVideo clips over TCP for bandwidth constrained conditions. Due to space constraints, we do not present these results here but refer the interested reader to [5].

5.2. TCP-(Un)Friendly

Although RealVideo over UDP may receive a disproportionate share of bandwidth versus their TCP counter-parts, this may be because RealVideo TCP clips transmit at less than their maximum rate. A more serious test of unfairness is whether RealVideo over UDP is *TCP-Friendly* in that its data rate does not exceed the maximum arrival of a conformant TCP connection in the same circumstances. The TCP-Friendly rate, T Bps, for a connection is given by [8]:

$$T \leq \frac{1.5\sqrt{2/3} \times s}{R \times \sqrt{p}} \qquad (1)$$

with packet size s, round-trip time R and packet drop rate p. For each clip for each run, we compute the TCP-Friendly rate (T) (equation 1), using a packet size (s) of 1500 bytes[12] and the loss rate (p) and RTT (R) obtained from the corresponding `ping` samples. We then compare T to the average bandwidth used by the UDP clip. For each bottleneck bandwidth, we record the count of the number of times the UDP clip was not TCP-Friendly.

Bottleneck Bandwidth	Total Unfriendly	$min>$ $fair$	$max<$ $fair$	Effective Unfriendly
75 Kbps	8/110 (7%)	22	30	8/58 (14%)
150 Kbps	7/110 (6%)	12	42	5/56 (9%)
300 Kbps	9/110 (8%)	12	48	7/50 (14%)
Total	24/330 (7%)	46	120	20/164 (14%)

Table 1. Number (and percent) of Non TCP-Friendly Flows

[12]The maximum packet size recorded. See [5] for more details on packet sizes.

The TCP-Friendly results are shown in Table 1[13]. The "Unfriendly" columns indicate a count of the UDP clips that were not TCP-Friendly. The "$min>fair$" column indicates the count of clips that had a minimum encoded bandwidth greater than the fair share of network bandwidth; these clips were not encoded to be able to properly respond to congestion. The "$max<fair$" column indicates the count of clips that had a maximum encoded bandwidth less than the fair share of network bandwidth; these clips, in general, had no need to respond to congestion. Removing the clips counted in these last two columns provides a base count for the non TCP-Friendly clips, presented in the column "Effective Unfriendly". This last analysis is useful as it exactly represents the percentage of RealVideo clips that must respond to congestion because of bandwidth constraints and have been encoded to allow the RealServer server to do so, but still behave in a non TCP-friendly manner.

Overall, 36% (120/330) of the UDP streams had a maximum bandwidth less than their fair share and thus were unconstrained by the network conditions. On the other hand, 14% (46/330) of the UDP streams were constrained by the network conditions but had not been encoded so as to allow them to respond to congestion. This latter set, while problematic from the congestion control point of view, can be readily addressed by content providers selecting multiple encoded bandwidths when creating streaming video content for their Web sites. Of the remaining UDP streams that were constrained by the network and had been encoded to allow a congestion response, 14% were not TCP-Friendly. Thus, with the proper bandwidth encoding levels (see Section 5.3), the large majority (86%) of RealVideo streaming video over UDP is TCP-friendly in the presence of network congestion.

The TCP-Friendly formula in equation 1 is conservative in that it computes the maximum bandwidth an aggressive TCP connection would receive. Thus, connections that achieve more bandwidth than computed in equation 1 are clearly not TCP-Friendly. In general, there is evidence to suggest many cases where streaming RealVideo over UDP is, in principle, TCP-Friendly, and there is also evidence to suggest that streaming RealVideo clips over UDP can sometimes be non TCP-Friendly, particularly for bandwidth constrained conditions.

5.3. Media Scaling

Media scaling technologies adapt media encoding to the available bandwidth in an effort to provide acceptable media quality over a range of available bandwidths [1, 22]. In times of congestion, media scaling benefits both the network, by reducing offered load, and also the user, by providing graceful degradation in perceived quality [21]. As

[13]Since the 600 Kbps bottleneck bandwidth clips had very low loss rates, we do not include the 600 Kbps data in our analysis to avoid data skew from "unlucky" sampling.

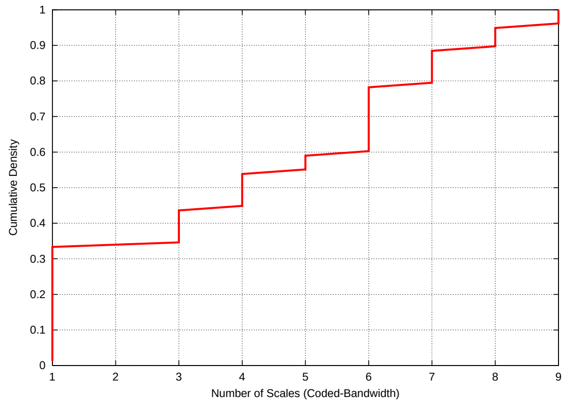

Fig. 5. CDF of Media Scales (All Runs)

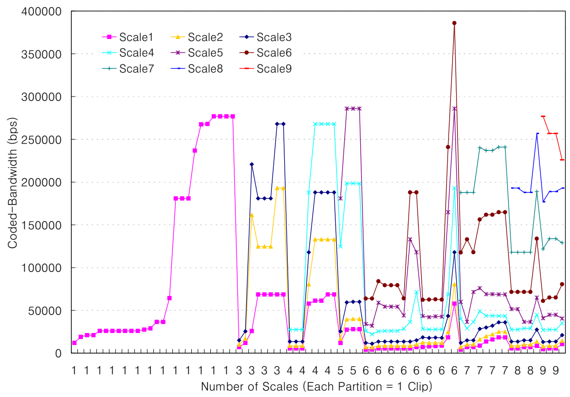

Fig. 6. Media Scales and Encoded-Bandwidth (All Clips). The horizontal-axis represents the number of different media scaling levels the clip can provide while the vertical axis represents the encoded bandwidth for each scale level. The clips are sorted from the fewest scales on the left to the most scales on the right. For ties, the clips with the lowest encoded bandwidth appear first.

mentioned in Section 2, RealSystems provide SureStream media scaling at the application level that can select an adequate quality version of a video to fit into the current network bandwidth conditions.

In the previous section, we showed that even if using media scaling, RealVideo streaming over UDP can be still non TCP-Friendly. This section analyzes data from the media scaling measurement experiments, as described in Section 3.3, in an effort to determine why.

Figure 5 shows a CDF of the number of distinct encoded-bandwidth levels seen in each clip for all runs. About 35% of the clips were not using media scaling at all, and therefore over UDP, these clips would be unresponsive to network congestion. Less than 50% of the clips were using more than 4 levels of scaling and so could only adjust to the available bandwidth coarsely.

Figure 6 shows the scale levels and corresponding bandwidths for each clip, sorted first by number of levels, and second by the lowest encoded bandwidth. For the unresponsive clips (those with only 1 scale level), 40% were high-quality video clips that required more than 150 Kbps of bandwidth. Also, over 50% of the clips with 3 to 5 scale levels were targeted primarily for broadband connections and could not adapt to bandwidths below 50 Kbps. Streaming these clips on bandwidth constrained links using UDP would cause unfairness to any competing TCP flows. RealVideo clips with more than 5 scale levels were designed to adapt more readily to low bandwidth conditions, evidenced by the number of scale levels with low bandwidths, but may still have been unfair at higher bandwidths.

When bandwidth is reduced during congestion, real-time streaming servers must employ media scaling in order to preserve timing, whether streaming over UDP or TCP. Figure 7 shows the media scaling behavior of two sample RealVideo clips streaming over UDP and TCP, where the inbound bandwidth available was 35 Kbps. For both clips and both streams, the initial encoded bandwidth was significantly higher than the available bandwidth, depicted by the horizontal line at 35 Kbps. Each horizontal "step" represents an application layer scaling of bandwidth. In the top graph of Figure 7, both TCP and UDP scaled their application data rate 6 times before the encoded rate settled at a proper application rate below the available bandwidth. However, UDP was able to obtain this application level rate much more quickly than did TCP. In the bottom graph of Figure 7, UDP quickly used 7 scale levels to adjust the application's data rate to the available bandwidth, while TCP, on the other hand, took more than 20 seconds to adjust the rate, and then it did so in one, large encoding rate change.

We believe the difficulty RealPlayer over TCP has in adjusting the application data rate to the network data rate is because TCP hides network information. Streaming applications over TCP can only measure application level goodput and not information on packet drop rates or network packet round-trip times. Streaming applications over UDP, on the other hand, can more easily detect packet losses and measure round-trip times, allowing them to more quickly adjust the application data rate to the network rate.

Moreover, for high-quality, high-bandwidth videos, the inability to detect network congestion when using TCP is critical. As evidenced by the TCP stream in the bottom graph of Figure 7, the server fills the available TCP buffers with high quality video frames that must be delivered by the transport layer before it is able to scale down. For the user, this results in a large delay before frame playout begins as the high-quality frames are buffered over a low-bandwidth connection. Quantitatively, by looking at the end-time of transmission, the top graph of Figure 7 shows that to play

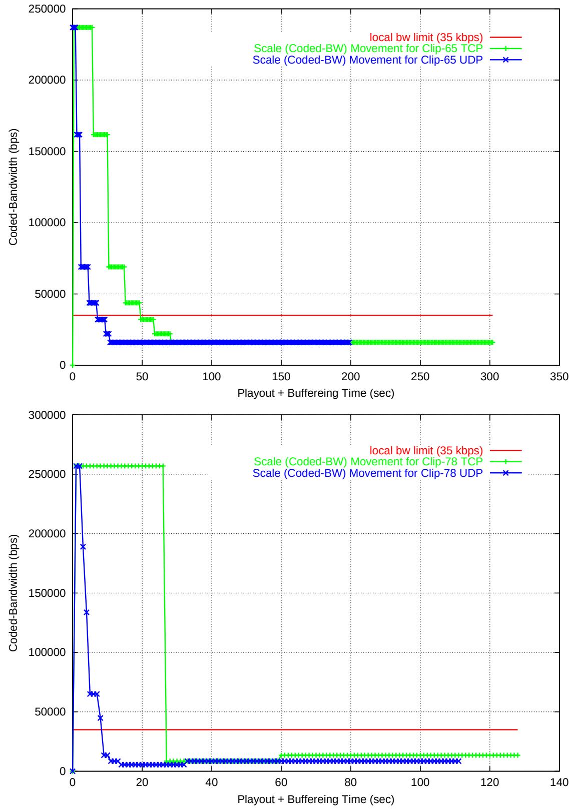

Fig. 7. Media Scaling Dynamics: Clip-65 (top) and Clip-78 (bottom) (DSL: BW=35 Kbps, Q=5 Kbytes)

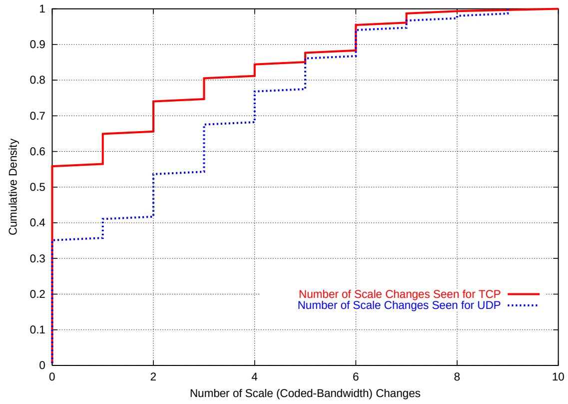

Fig. 8. CDF of Media Scale Changes (DSL: BW=35 Kbps, Q=5 Kbytes)

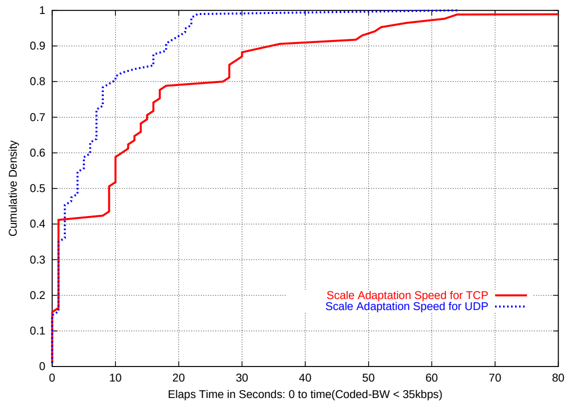

Fig. 9. CDF of Media Scale Adaptation Speed (DSL: BW=35 Kbps, Q=5 Kbytes)

3 minutes of video, streaming over UDP took about 200 seconds while streaming over TCP took more than 300 seconds. In other words, streaming over UDP required 20 seconds of buffering to play 3 minutes of a video clip, while streaming over TCP required more than 2 minutes of buffering to play the same clip.

In Figure 8, the CDFs depict the number of media scale changes seen for each video clip, and summarize the relative responsiveness of RealVideos to scale the application data rate to below the network bandwidth. Overall, UDP streams had more scale changes than did TCP streams. Also, Figure 8 shows that about 20% (55% - 35%) of the streams that scaled when streamed over UDP did not scale at all when streamed over TCP.

Figure 9 summarizes the responsiveness of RealVideo media scaling based on how quickly the video stream adapted to the available bandwidth after streaming started. Specifically, for the successfully adapted streams, we measure the time taken for the coded-bandwidth to drop under the inbound bandwidth limit, depicted as the first point under the 35 Kbps limit for each stream in Figure 7. Figure 9 shows that about 15% of video clips were low-quality and always required less than 35 Kbps. Also, 25% (40% - 15%) of the video clips were able to adapt to the available bandwidth within a couple of seconds, independent of the transport protocol used. However, for the remaining 60% of the clips, the TCP video streams took significantly more time to adapt their scales to the available bandwidth. For example, 80% of the UDP video streams adapted to the available bandwidth within 10 seconds, while it took more than 25 seconds for the same percentage of the TCP video streams to adapt.

In general, a significant fraction of RealVideo clips are unable to adapt their application data rates to the available network bandwidth, causing UDP streaming to be unfair under bandwidth constrained conditions. However, most RealVideo clips can, and do, scale their application data rates to the available network bandwidth. RealVideo streams over UDP can adjust their application data rates to the available bandwidth more efficiently than can RealVideo over TCP.

5.4. Buffering Data Rate

As shown in [11], RealPlayer buffers data at an accelerated rate for the first part of a clip. Analyzing the rate of this

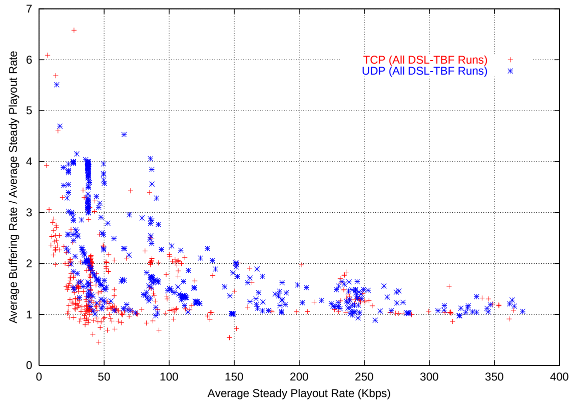

Fig. 10. Ratio of Average Buffering Rate to Average Steady Playout Rate versus Average Steady Playout Rate (All Runs)

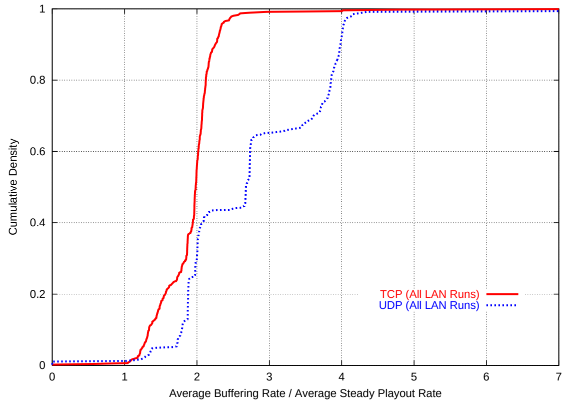

Fig. 11. CDF of Ratio of Average Buffering Rate to Average Steady Playout Rate (LAN)

buffering rate versus steady playout rate may help to characterize the bursty nature of RealVideo streams.

For each clip, we compute the maximum bandwidth averaged over 10 second intervals taken over the first 80 seconds (calling this the *buffering data rate*) and compared this to the average bandwidth over the time from 100 seconds until the clip ends (calling this the *steady playout rate*).

Figure 10 depicts the ratio of the average buffering data rate to the average steady playout rate for different steady playout rates. For reference, a ratio of 1 indicates that the buffering data rate was equivalent to the steady playout rate. From Figure 10, low bandwidth clips buffered at up to 6 times their average playout rate. Higher bandwidth clips buffered at relatively lower rates, possibly because total bandwidth restrictions limited them from buffering at a higher rate.

In order to determine if bandwidth restrictions limit buffering rates, we ran a set of experiments with the bottleneck bandwidth being the campus LAN attached to the Internet via a 15 Mbps link[14]. In this setup, the LAN environment was relatively unconstrained, having a bottleneck bandwidth which was typically at least three times that of our 600 Kbps bottleneck bandwidth.

Figure 11 depicts a CDF of the ratio of the average buffering data rate to the average steady playout rate. The ratio of buffering rate to steady rate for UDP was nearly the same as that of TCP for 40% of the clips. For 60% of the clips, however, the ratio of buffering rate to steady for UDP was significantly higher than that of TCP. For UDP, the vertical "steps" in the CDF are at typical RealVideo bandwidth encoding rates, where the buffering rate was a fixed multiple of these rates. For TCP, the steep slope in the CDF at around 2 suggests TCP streams typically buffered at a rate twice that of the steady playout rate.

[14]http://www.wpi.edu/Admin/Netops/MRTG/

In general, both RealVideo clips over UDP and RealVideo clips over TCP buffer data at a significantly higher rate than the steady playout rate, suggesting that overall RealVideo traffic is bursty over the length of the clip and not strictly constant bitrate (CBR).

6. DISCUSSION OF RESULTS

In the current Internet, there are no concrete incentives for applications that use UDP to initiate end-to-end congestion control. In fact, at the network level, unresponsive applications may be "rewarded" by receiving more than their fair share of link bandwidth. As seen in Section 5, streaming media over UDP can result in a higher average bandwidth rate than streaming media over TCP, primarily because competing TCP sources are forced to transmit at a reduced rate. Plus, as seen in Section 5.3, it is more difficult for the application layer to adjust the encoding rate to the available bandwidth when using TCP (because there is no API that gives you available bandwidth, for example). Thus, there are strong application-oriented reasons for streaming media to use UDP rather than TCP, suggesting potentially high-bandwidth video over UDP may contribute to congestion collapse.

However, given the current climate where it is recognized that end-to-end congestion control, and even TCP-Friendly congestion control, is fundamentally important to the well-being of the Internet, there are likely social pressures for video software designers not to release products without some form of end-to-end congestion control. Moreover, an unresponsive "fire-hose" application, such as high quality video over a congested link, is ineffective from the application standpoint primarily because having a congested router randomly drop packets can cause the more important data packets to be dropped. Instead, applications can significantly benefit by using media scaling, as illustrated by RealPlayer in Section 5.3, to make intelligent decisions about which packets not to send beforehand, making low quality video over the same congested link quite effective. Anecdotally, in our pilot tests with severe congestion, older versions of RealPlayer would continue to attempt to stream video, inducing even more congestion, while newer versions of RealPlayer would terminate the connection under the same conditions. Moreover, as shown in Section 5.3, RealVideo over UDP clearly scales the application data rate to meet the available bandwidth. Thus, while it is not clear as to exactly what degree practical or social incentives are effective, there is evidence to suggest they are having a significant impact.

The higher buffering rate seen in Section 5.4 is beneficial for users, but possibly harmful to the network. A higher buffering rate either allows the player to build up a larger buffer before beginning frame playback and thus better avoiding any unsmoothness caused by network jitter or transient congestion, or allows the frame playback to begin earlier. However, the increased buffering rate makes the streaming traffic more bursty and, with UDP, it can cause even more unfairness versus other TCP flows. Overall, from the network point of view, the buffering rate should be limited to the playout rate, and is so in some other commercial players [11].

7. CONCLUSIONS

The decreasing cost of powerful PCs and the increase in video content on the Web is fueling the growth of streaming video over the Internet. Unlike traditional applications, streaming video often uses UDP as a transport protocol rather than TCP, suggesting that streaming video may not be TCP-friendly or, worse, that streaming video may be unresponsive to network congestion. Since congestion control is fundamentally important to the health of the Internet, a better understanding of the responsiveness (or lack of it) of streaming video using UDP can help focus network layer research that detects and polices unresponsive flows, or transport layer research that develops better streaming protocols.

Commercial streaming video players, such as RealNetworks' RealPlayer, promise to have a large influence on the impact of streaming video on the Internet. While previous empirical studies have focused on Internet traffic in general or have concentrated on overall measurements of streaming applications, to the best of our knowledge, there have been no detailed studies on the responsiveness to congestion of commercial players streaming over UDP.

In this work, we evaluate the network-level and application-level responsiveness of RealVideo streaming over UDP by comparing it to TCP under the same network conditions. We set up a testbed that allows us to simultaneously stream two RealVideo clips, one over TCP and one over UDP, along the same network path. Our testbed also lets us control the network bottleneck bandwidth, thus allowing us to evaluate the responsiveness to congestion of the UDP streams. Using our testbed, we stream over 600 hours of videos from over 1000 video clips with a variety of content and encoding bandwidths selected from across the Internet.

Overall, we find RealVideo over UDP typically receives the same bandwidth as that of TCP. Even during periods of packet loss, most RealVideo over UDP is TCP-Friendly. However, under very constrained bandwidth conditions, RealVideo over UDP can get substantially more bandwidth than TCP and the bandwidth use gets increasingly unfair with an increase in packet loss rate and round-trip time.

Most RealServers can, and often do, scale the application layer data rate in an attempt to match the network data rate. Application scaling tends to be coarser at higher levels of bandwidth but is often fine grained at lower levels of bandwidth. While application scaling can be an effective means of responding to congestion, about 35% of RealVideos cannot do application scaling at all, making them unresponsive to network congestion when streaming over UDP. Adjusting the application data rate to the network bandwidth is more difficult when streaming over TCP versus UDP, most likely because application streams over TCP do not have as much information about the current network state as do the application streams over UDP.

RealPlayers typically buffer video data for up to 40 seconds at a much higher rate than the average playout rate. While beneficial to the user, this initial burst of traffic can cause considerable congestion and probably makes RealVideo network traffic more difficult to manage.

8. FUTURE WORK

This work is only another step in the analysis of streaming multimedia traffic on the Internet, leaving many areas for future work.

The major commercial competitor to RealNetworks' RealPlayer is Microsoft's Windows Media Player[15]. Measurement of the congestion responsiveness of Media Player streaming over UDP might help understand the differences in congestion responsiveness across commercial players. We have conducted preliminary comparisons of RealPlayer and Media Player in [11].

We intentionally selected pre-recorded video clips to help ensure consistency in the videos played out during each set of experiments. Live content, captured and served directly from a video camera or television, typically has different characteristics than does pre-recorded content. Future work could be to measure the performance of live RealVideo content on the Internet and compare it to that of the pre-recorded RealVideo content in our study.

[15] http://www.microsoft.com/windows/windowsmedia/default.asp

The work in this paper did not explore the relationship between perceptual quality of the video, influenced by application level metrics such as frame rate and jitter, and network metrics. A better understanding of the impact on perceptual quality on video streaming over UDP versus TCP might further aid in developing more effective ways to use a TCP-Friendly share of bandwidth.

9. REFERENCES

[1] P. Bocheck, A. Campbell, S.-F. Chang, and R. Lio, "Utility-based Network Adaptation for MPEG-4 Systems," in *Proceedings of International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*, June 1999.

[2] J.-C. Bolot, S. Fosse-Parisis, and D. Towsley, "Adaptive FEC-Based Error Control for Internet Telephony," in *Proceedings of IEEE INFOCOM*, Mar. 1999.

[3] Z. Cao, Z. Wang, and E. Zegura, "Rainbow Fair Queuing: Fair Bandwidth Sharing Without Per-Flow State," in *Proceedings of IEEE INFOCOM*, Mar. 2000.

[4] M. Chesire, A. Wolman, G. Voelker, and H. Levy, "Measurement and Analysis of a Streaming Media Workload," in *Proceedings of the USENIX Symposium on Internet Technologies and Systems (USITS)*, Mar. 2001.

[5] J. Chung, Y. Zhu, and M. Claypool, "FairPlayer or FoulPlayer? - Head to Head Performance of RealPlayer Streaming Video Over UDP versus TCP," CS Department, Worcester Polytechnic Institute, Tech. Rep. WPI-CS-TR-02-17, May 2002.

[6] G. J. Conklin, G. S. Greenbaum, K. O. Lillevold, A. F. Lippman, and Y. A. Reznik, "Video Coding for Streaming Media Delivery on the Internet," *IEEE Transactions on Circuits and Systems*, pp. 269 – 281, Mar. 2001.

[7] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness," in *Proceedings of IEEE INFOCOM*, Apr. 2001.

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

[9] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," in *Proceedings of ACM SIGCOMM Conference*, 2000, pp. 45 – 58.

[10] Jupiter Media Metrix, "Users of Media Player Applications Increased 33 Percent Since Last Year," Apr. 2001, Press Release. http://www.jup.com/company/pressrelease-.jsp?doc=pr01040.

[11] M. Li, M. Claypool, and R. Kinicki, "MediaPlayer versus RealPlayer – A Comparison of Network Turbulence," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, Nov. 2002.

[12] Y. Liu and M. Claypool, "Using Redundancy to Repair Video Damaged by Network Data Loss," in *Proceedings of IS&T/SPIE/ACM Multimedia Computing and Networking (MMCN)*, Jan. 2000.

[13] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling High-Bandwidth Flows at the Congested Routers," in *In Proceedings of the 9th International Conference on Network Protocols (ICNP)*, Nov. 2001.

[14] A. Mena and J. Heidemann, "An Empirical Study of Real Audio Traffic," in *In Proceedings of the IEEE Infocom*, Mar. 2000, pp. 101 – 110.

[15] D. Mitra, K. Stanley, R. Pan, B. Prabhakar, and K. Psounis, "CHOKE, A Stateless Active Queue Management Scheme for Approximating Fair Bandwidth Allocation," in *Proceedings of IEEE INFOCOM*, Mar. 2000.

[16] K. Park and W. Wang, "QoS-Sensitive Transport of Real-Time MPEG Video Using Adaptive Forward Error Correction," in *Proceedings of IEEE Multimedia Systems*, June 1999, pp. 426 – 432.

[17] Real Networks Incorporated, "RealNetworks Facts," 2001, URL: http://www.reanetworks.com/gcompany/index.html.

[18] Real Networks Incorporated, "RealProducer User's Guide," copyright 2000, URL: http://www.service.real.com/help/library/guides-/producerplus85/producer.htm.

[19] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Real-time Streams in the Internet," in *Proceedings of IEEE Infocom*, 1999.

[20] I. Stoica, S. Shenker, and H. Zhang, "*Core*-Stateless Fair Queueing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," in *Proceedings of ACM SIGCOMM Conference*, Sept. 1998.

[21] A. Tripathi and M. Claypool, "Improving Multimedia Streaming with Content-Aware Video Scaling," in *Workshop on Intelligent Multimedia Computing and Networking (IMMCN)*, Mar. 2002.

[22] J. Walpole, R. Koster, S. Cen, C. Cowan, D. Maier, D. McNamee, C. Pu, D. Steere, and L. Yu, "A Player for Adaptive MPEG Video Streaming Over The Internet," in *Proceedings of the SPIE Applied Imagery Pattern Recognition Workshop*, Oct. 1997.

[23] Y. Wang, M. Claypool, and Z. Zuo, "An Empirical Study of RealVideo Performance Across the Internet," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, Nov. 2001.