Empirical Evaluation of the Congestion Responsiveness of RealPlayer Video Streams

Jae Chung and Mark Claypool
Computer Science Department
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609, USA
{goos|claypool}@cs.wpi.edu

***Abstract*— Increasingly popular commercial streaming media applications over the Internet often use UDP as the underlying transmission protocol for performance reasons. 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. Unfortunately, there are few empirical studies that analyze the responsiveness, or lack of it, of commercial streaming media applications. In this work, we evaluate the responsiveness of RealNetworks' RealVideo over UDP by measuring the performance of numerous streaming video clips selected from a variety of RealServers on the Internet, analyze the TCP-Friendliness of the UDP streams and correlate the results with network and application layer statistics. We find that 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 share of the capacity often achieve a TCP-Friendly rate. In addition, our results suggest 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 capacity for effective media scaling.**

***Index Terms*— streaming media, protocols, RealPlayer, TCP-Friendly, unresponsive flows**

I. 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. For example, in 2001 an estimated of 350,000 hours of online entertainment was broadcast each week over the Internet [25], with countless more hours downloaded on-demand. Web sites today offer streaming videos of news broadcasts, music television, live sporting events and more.

While voice quality audio typically operates over a narrow range of bitrates (32-64 Kbps), video operates over a much wider range of bitrates. 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]. Thus, video applications have the potential to demand enormous bitrates, often greater than the available network capacity, but also have the potential to reduce their data rates when available capacity is constrained.

While TCP is the de facto standard transport protocol for typical Internet application protocols such as HTTP, FTP and SMTP, 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 practitioners typically prefer 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 [28], [11], but such protocols are not yet widely part of most operating system distributions. For these reasons, streaming video applications often use UDP as a transport protocol rather than TCP. Moreover, with the use of repair techniques [3], [18], [23], [24], UDP 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 applications to lower their bitrate in the presence of packet loss during congestion.

Potentially high-bitrate video over UDP using repair

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

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 video flows using UDP reduce their sending rates in response to congestion, leaving the unresponsive UDP flows to expand to use the vacant capacity, or, worse, contribute to congestion collapse of the Internet [10].

In light of this, recent research has explored router queue management approaches to identify and police unresponsive flows [19], [29], [9], [17], [21], [5]. Such research often models unresponsive flows as transmitting data at a constant packet size and constant packet rate (CBR), or as "fire-hose" applications, transmitting at an unyielding, maximum rate. However, commercial media products have been shown to not be strictly CBR [20], [16], [22], 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 [13]. 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 [20], [31], [6], [16], [34], as well as some measurements of commercial streaming media on private testbeds [14], [22], there are few empirical studies that analyze the responsiveness, or lack of it, of current streaming media products over the Internet.

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 bitrate to the clients by limiting the capacity 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 bitrates 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 capacity-constrained for a typical broadband connection, resulting in a fair share of link capacity for a RealVideo stream over UDP and a single TCP flow. In cases with reduced capacity, most streaming RealVideo over UDP does respond by reducing the application layer encoding rate, often responding to meet TCP-Friendly criteria. 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 II presents background on RealPlayer to help understand our results; Section III describes our approach to obtain a wide-range of Internet measurements; Sections IV and V present and analyze, respectively, the measurement data obtained; Section VI discusses our findings; Section VII summarizes our conclusions and Section VIII presents possible future work.

II. RealVideo Background

RealPlayer provided by RealNetworks,[4] is one of the most popular streaming media players on the US Internet, with over 47% of the commercial market share in April 2001 [13]. 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 encoding bitrates appropriate for their target audience and specify other encoding parameters, such as frame size and frame rate, appropriate for their content. The RealServer then streams the video to a user's RealPlayer client upon request.

RealServer and players primarily use the 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. Older versions of RealServer used the Progressive Networks Audio (PNA) protocol and, for backward compatibility, newer RealServers and players still support this protocol. For this measurement study, all the video clips selected used RTSP, as described in Section III-A.

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

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

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 [34]. The decision making process RealPlayer uses to choose either UDP or TCP is not publicly documented, and may be interesting future work. The choice of UDP or TCP can also be manually specified by the user [26]. For our study, we specifically set RealPlayer to use UDP in some cases and TCP in others, as described in Section III-B.

RealSystem supports an application level media scaling technology called *SureStream* in which a RealVideo clip is encoded for multiple target bitrates [8], [27]. When streaming a SureStream RealVideo clip, RealServer determines which encoded stream to use based on feedback from the RealPlayer regarding the client end-host network conditions. With SureStream, the actual video stream served can be varied in mid-playout, with the server switching to a lower bitrate stream during network congestion and then back to a higher bitrate stream when congestion clears. We study the flexibility of SureStream scaling in Section V-C.

For each video clip, RealPlayer keeps a buffer to smooth out the video stream because of changes in capacity, 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 available capacities 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 V-D.

III. Approach

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

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

A. 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. If we had chosen a stand-alone environment where 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 selected 100 valid RTSP RealVideo URLs from the first 100 search results returned, of which 79 were available during the experimental runs.

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 IV, Section V-C and [7].

B. 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 network 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 lower bitrate than a bulk TCP transfer under the same network conditions, providing a "lower bound" on the bitrate a bulk TCP transfer would use.

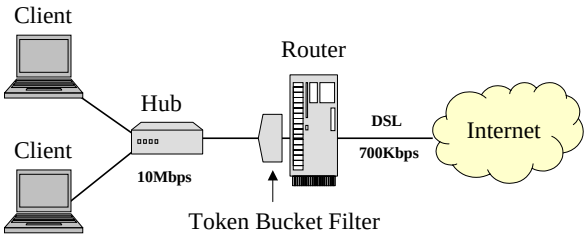

Fig. 1. Testbed Network Setup: Environment to Measure the Responsiveness of RealVideo

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

The hub facilitated capturing network layer performance data since packets destined to either PC were broadcasted to both PCs. We ran `tcpdump`[6], 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`[7] 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 the playout of 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 the `ping` samples were comparable to those obtained via `tcptrace`. Also, we verified that the packet filtering and logging did not induce significant 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 `tcpdump` 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 available bitrate, we set up a private Linux router connected to a commercial 700 Kbps DSL network to enable us to create constrained bitrate 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 Linux token bucket filter (TBF)[9] module 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 bitrate) 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 capacity share is approximately half of each bottleneck capacity.

Although it is conventional wisdom that overprovisioning in core network routers has moved Internet performance bottlenecks to network access points [1], it is still possible that network bottlenecks may occur elsewhere. However, for our study, the location of the bottleneck, whether at the access link or further upstream, does not impact the competition between the TCP and UDP streams since the streams have the same NAT-translated IP address and thus share the same network path. Even if the network path is altered mid-stream due to a routing change, the change will be applied to both streams.

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

C. Media Scaling Measurement Environment

Streaming video can adjust to the available capacity during congestion by *media scaling* where video encoding is switched to a lower rate. As mentioned in Section II, RealSystems uses a media scaling technology called *SureStream* in which a RealVideo clip is encoded for multiple bitrates [27]. The actual video stream served can be varied in mid-playout, with the server switching to a lower bitrate stream during network congestion and then back to a higher bitrate stream when congestion clears.

To study media scaling in RealPlayer we used *RealTracer*,[10] developed for a previous study [34], which plays RealVideo streams and records application level statistics, including encoding rate. One of the client machines was

[6] http://www.tcpdump.org/
[7] http://irg.cs.ohiou.edu/tcptrace/tcptrace_new/
[8] http://www.ethereal.com/

[9] Recent work measuring broadband access links suggest some ISP providers similarly use TBFs to limit capacities [15].
[10] http://perform.wpi.edu/real-tracer/

booted to run 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 bitrate to 35 Kbps,[11] since the highest encoded bitrate 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 bitrates) 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 was used to characterize the responsiveness of RealVideo media scaling (see Section V-C).

IV. Results

Over the course of 2 months, we streamed a total of over 200 hours of video from a cumulative total of over 2000 video clips. Of the total 79 clips in the playlist, 24 (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, loss and TCP-Friendly analysis in this report, we removed the data from these clips, leaving a total of 110 clips for each protocol type at each bottleneck capacity (55 clips $\times$ 8 $\times$ 2 sets of experiments = 880 total clips). However, we did use the other data recorded on the set of 148 clips for each protocol type at each bottleneck capacity (1184 clips total) for analysis that did not require use of the `ping` data.

Comparing the average RTTs obtained via `ping` probes for each bottleneck capacity, 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 about the same RTT regardless of the bottleneck capacity since these clips streamed 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 capacity 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 capacity, the median loss rate for any configuration was less than 2%. About 37% of the clips played with low bottleneck capacities had no loss, while about 50% of the clips played at higher bottleneck capacities had no loss. Overall loss rates increased about 1% for each decrease in bottleneck capacity from 600 Kbps to 300 Kbps to 150 Kbps to 75 Kbps. The low loss rates, even at low capacities, implies that most of the RealVideo UDP streams adapted to the available bitrate, and is investigated in depth in Section V.

Summarizing statistics on packet sizes, the TCP streams used larger packets than the UDP streams with a median UDP packet size of about 640 Kbytes, and a median TCP packet size of about 1100 Kbytes. Moreover, more than 30% of the TCP packets were equal to the typical network MTU, 1500 Bytes. A possible reason for the larger packet sizes over TCP is that while RealServers can control the application frame sizes to send, with TCP, those frames are often grouped and sent based on the current TCP window sizes.

We do no present further details on the results here, but refer the interested reader to [7].

V. Analysis

In analyzing the responsiveness of RealVideo over UDP, we first analyze bitrates aggregated over all clips and then analyze bitrates for individual clip pairs (Section V-A). We next analyze the TCP-Friendliness of RealVideo over UDP (Section V-B). Moving to the application layer, we analyze the application scaling behavior (Section V-C). Lastly, we measure the initial buffering rate compared with the steady playout rate (Section V-D).

A. Bitrates

Figure 2 depicts Cumulative Density Functions (CDFs) of the per-clip average bitrate used by TCP and UDP for bottleneck capacities of 600, 300, 150 and 75 Kbps. The TCP and UDP distributions are nearly the same for the 600 Kbps bottleneck capacity. However, as capacity becomes more constrained, the distributions separate, with UDP having a consistently higher distribution of bitrates than TCP, as evidenced by UDP distributions being lower and to the right of the corresponding TCP distributions.

We next analyze the head-to-head bitrate 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 bitrate used by the TCP stream and y is the average bitrate used by the UDP stream. The points for each bottleneck capacity are depicted by a different point style. The dashed 45 degree line provides a reference for equal bitrates for both TCP and UDP. Points above the line (top left of the graph) indicate UDP had a higher average bitrate while points below the line (bottom right of the graph) indicate TCP had a higher average bitrate. The distance from the line indicates the magnitude of the average bitrate difference.

[11] The queue was set to 5 Kbytes for the 35 Kbps DSL-TBF configuration.

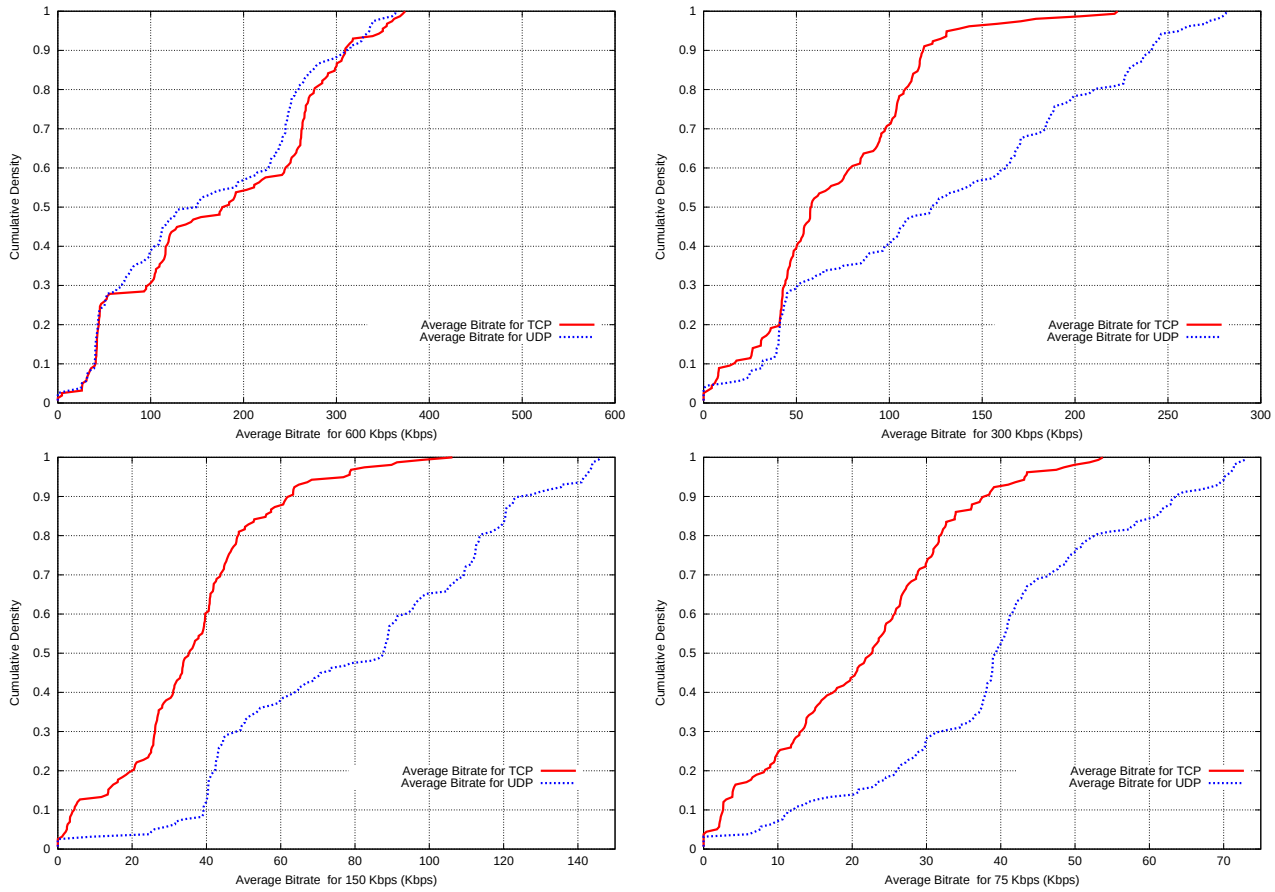

Fig. 2. CDFs of Average Bitrates for Bottleneck Capacities of 600, 300, 150, and 75 Kbps

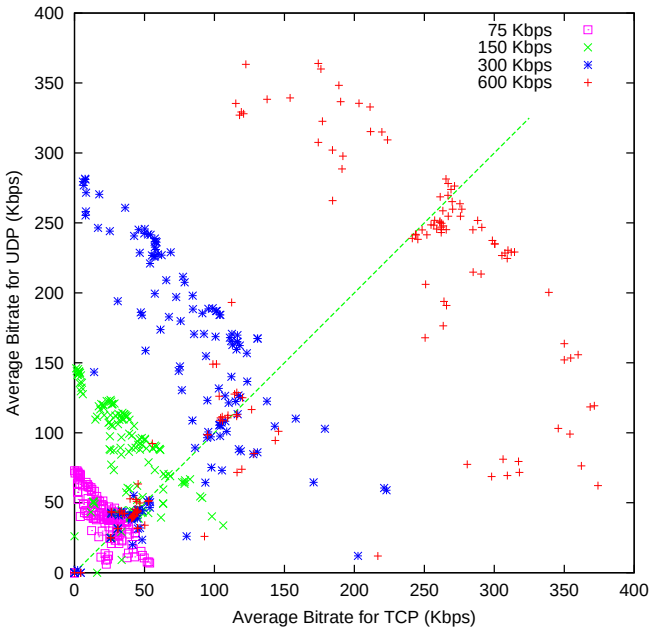

Fig. 3. Head-to-Head Average Bitrate (all runs)

From Figure 3, while there are some points that lie along the equal bitrate line, there are many cases of bitrate disparity. The highest bitrate playouts for the 600 Kbps bottleneck capacity had the greatest bitrate disparities. For the 600 Kbps bottleneck capacities, there are visually as many points below the equal bitrate line where TCP had a higher average bitrate as there are above the equal bitrate line where UDP had a higher average bitrate. For the lower bottleneck capacities, there are visually considerably more points above the equal bitrate line, indicating UDP had a higher average bitrate than did TCP.

We next analyze the bitrate disparity relative to the bottleneck capacity. For each clip pair, we subtract the UDP average bitrate from the TCP average bitrate and divide the difference by the bottleneck capacity. Thus, equal sharing of capacity has a value of 0, a value of -1 indicates UDP got the entire bottleneck capacity, and a value of +1 indicates TCP got the entire bottleneck capacity. Figure 4 depicts CDFs of the normalized bitrate differences for each bottleneck capacity.

For the 600 Kbps bottleneck capacity, about 40% of the clips shared the capacity equally. As indicated by the region in the top right, about 30% of TCP clips had a higher bitrate than their counterpart UDP clips while about 20% of the UDP flows had a higher bitrate than their counterpart TCP clips, as indicated by the region in the bottom

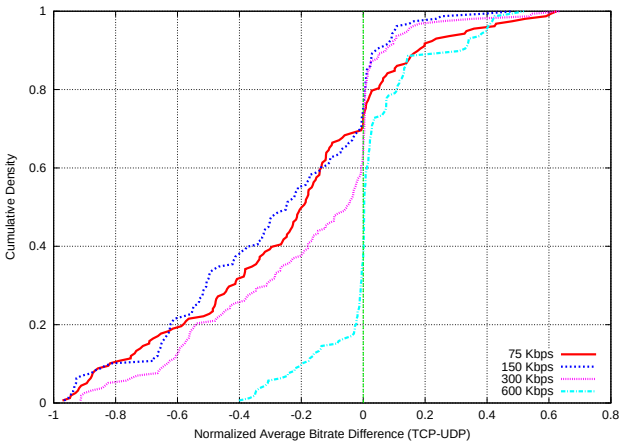

Fig. 4. CDF of the Difference (TCP - UDP) in the Average Bitrate, Normalized by the Bottleneck Capacity (all runs)

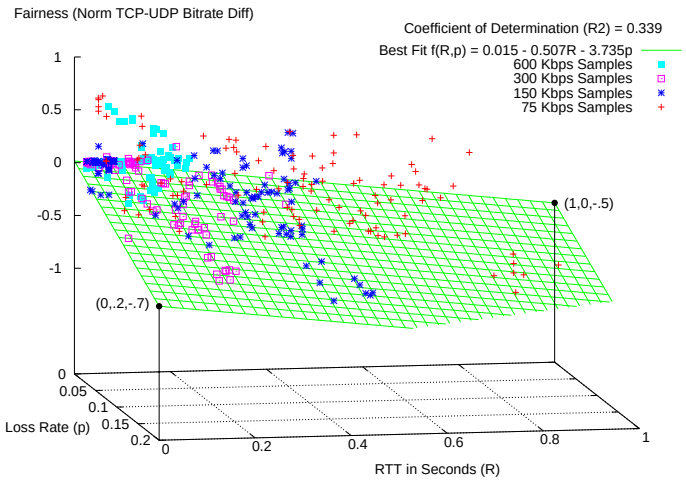

Fig. 5. Loss Rate, Round-Trip Time and Fairness (Normalized TCP-UDP Average Bitrate Difference)

left. For the 600 Kbps bottleneck capacity, the greatest bitrate disparity was approximately half the bottleneck capacity.

For the lower bottleneck capacities, there were increasingly fewer clips with equal bitrates. Many UDP clips had substantially higher bitrates than did their TCP counterparts, as indicated by the large areas under the distributions on the bottom left. For the 300 Kbps bottleneck capacity, about 60% of the UDP clips had higher bitrates than their TCP counterparts, and for the 150 Kbps and 75 Kbps bottleneck capacities, about 70% of the UDP clips had higher bitrates than their TCP counterparts. For the 300, 150 and 75 Kbps bottleneck capacities, about 20% of the UDP clips got twice the normalized bitrate of their TCP counterparts. For the 150 and 75 Kbps bottleneck capacities, about 20% of the UDP clips had more than 80% more of the normalized bitrate than their TCP counterparts. However, even for the lowest bottleneck capacities, there were still cases where the TCP clips had a higher bitrate than their UDP counterparts, as depicted by the areas above the distributions in the upper right.

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

We next analyze the impact of round-trip time and loss rate on the normalized bitrate disparity. The data rate of TCP is paced by acknowledgments and is limited by packet loss rate, so a higher round-trip time or loss rate directly results in a lower maximum throughput. However, the data rate of UDP is not similarly constrained. Our analysis of round-trip times and loss rates obtained by the `ping` samples show modest correlations for both round-trip times and normalized bitrate disparity and loss rates and normalized bitrate disparity. Figure 5 plots the normalized bitrate differences as a function of round-trip time and loss rate, and draws the best-fit (least square) plane for the samples. The coefficient of determination (R^2) of 0.339 indicates that the regression plane explains about one-third of the variation in the normalized bitrate disparity. The correlation of -0.51 for the round-trip time in seconds and -3.7 for the loss rate indicates that as round-trip times and loss rates increase, streaming RealVideo clips over UDP receive relatively more of the available bitrate than do streaming RealVideo clips over TCP.

B. *Discussion of TCP-(Un)Friendliness*

Although RealVideo over UDP may receive a disproportionate share of available bitrate than do their TCP counterparts, 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 rate of a conformant TCP connection under the same network conditions. The TCP-Friendly rate, T Bps, for a connection is given by [10]:

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

with packet size s, round-trip time R and packet drop rate p. For each clip for each run, we use Equation (1) to compute the TCP-Friendly rate (T), 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 bitrate used by the UDP clip. For each bottleneck capacity, we record the count of the number of times the UDP clip was not TCP-Friendly.

TABLE I
NUMBER (AND PERCENT) OF NON TCP-FRIENDLY FLOWS

Bottleneck Capacity	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%)

The TCP-Friendly results are shown in Table I[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 bitrate greater than the fair share of network capacity; 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 capacity less than the fair share of the available bitrate; 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 available bitrate constraints and have been encoded to allow the RealServer server to do so.

Overall, 36% (120/330) of the UDP streams had a maximum bitrate 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 bitrates 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 bitrate encoding levels (see Section V-C), the large majority (86%) of RealVideo streaming video over UDP is TCP-friendly in the presence of network congestion.

[12]The maximum packet size recorded. See [7] for more details on packet sizes.

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

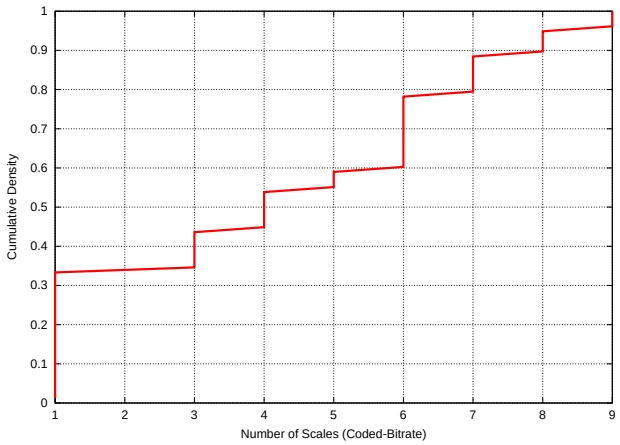

Fig. 6. CDF of Media Scales (all runs)

The TCP-Friendly formula in Equation (1) is conservative in that it computes the maximum bitrate an aggressive TCP connection would receive. Thus, connections that achieve a higher bitrate 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 capacity-constrained conditions.

C. Media Scaling

Media scaling technologies adapt media encoding to the available bitrate in an effort to provide acceptable media quality over a range of available bitrates [2], [33]. 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 [30]. As mentioned in Section II, RealSystems provide SureStream media scaling at the application level that can select an adequate quality version of a video to fit into the current conditions of available network bitrate.

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

Figure 6 shows a CDF of the number of distinct encoded bitrate 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 have difficulty responding 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 bitrate coarsely.

Figure 7 shows the scale levels and corresponding bitrates for each clip, sorted first by number of levels, and

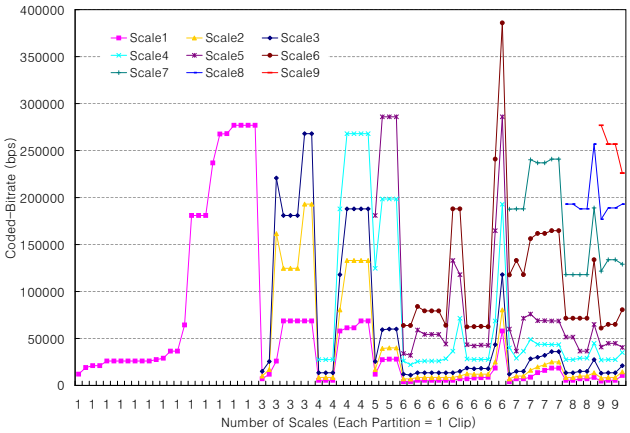

Fig. 7. Media Scales and Encoded-Bandwidth (all clips). The horizontal-axis represents the number of different media scaling levels the clip can provide while the vertical axis represents the encoded bitrate for each scale level. The clips are sorted from the fewest scales on the left to the most scales on the right. For ties, the clips with the lowest encoded bitrate appear first.

second by the lowest encoded bitrate. For the unresponsive clips (those with only 1 scale level), 40% were high-quality video clips that had a bitrate higher than 150 Kbps. Also, over 50% of the clips with 3 to 5 scale levels were targeted primarily for broadband connections and could not adapt to capacities below 50 Kbps. Streaming these clips on capacity-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 capacity conditions, evidenced by the number of scale levels with low bitrates, but may still be unfair at higher capacities.

When available bitrate is reduced during congestion, real-time streaming servers must employ media scaling in order to preserve timing, whether streaming over UDP or TCP. Figure 8 shows the media scaling behavior of two sample RealVideo clips streaming over UDP and TCP, where the available inbound bitrate was 35 Kbps. For both clips and both streams, the initial encoded bitrate was significantly higher than the available capacity, depicted by the horizontal line at 35 Kbps. Each horizontal "step" represents an application layer scaling of bitrate. The final playout bitrates achieved show the conservative adaptation of the RealServers since they stabilize at a bitrate less than what is available. This conservative media scaling behavior may result in less than optimal video quality but often helps the UDP RealVideo streams to achieve TCP-Friendly rates as supported by our TCP-Friendliness analysis. In the top graph of Figure 8, both TCP and UDP scaled their application data rate 6 times before the encoded rate settled below the available bitrate. However, UDP was able to obtain this application data rate much

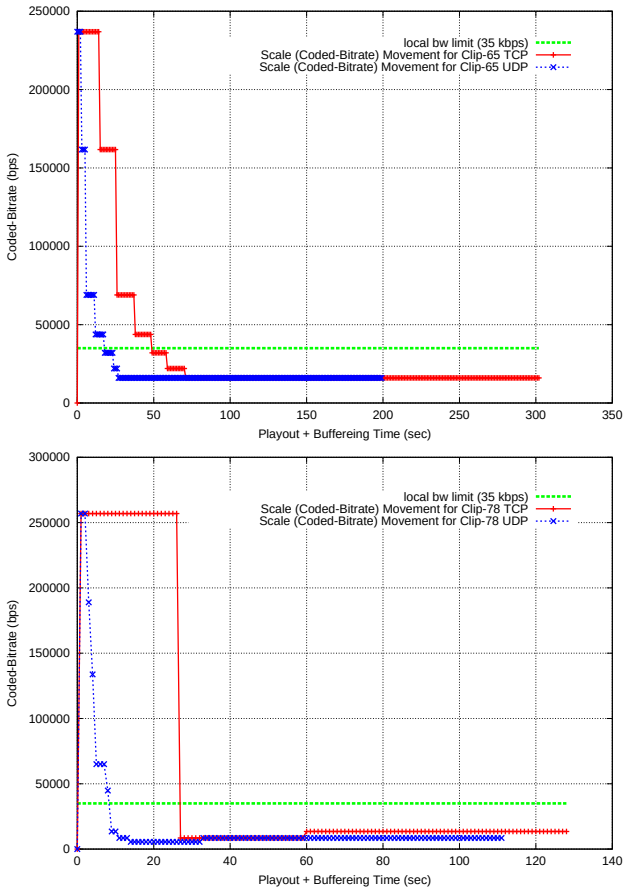

Fig. 8. Media Scaling Dynamics: Clip-65 (top) and Clip-78 (bottom) (DSL: BR=35 Kbps, Q=5 Kbytes)

more quickly than did TCP. In the bottom graph of Figure 8, UDP quickly used 7 scale levels to adjust the application's data rate to the available bitrate, 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 available network bitrate is because TCP does not expose network information to the application layer. Streaming applications over TCP can only measure application level goodput and not information on packet drop rates or 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 available network bitrate.

Moreover, for high-quality, high-bitrate videos, the inability to detect network congestion when using TCP is critical. As evidenced by the TCP stream in the bottom graph of Figure 8, 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

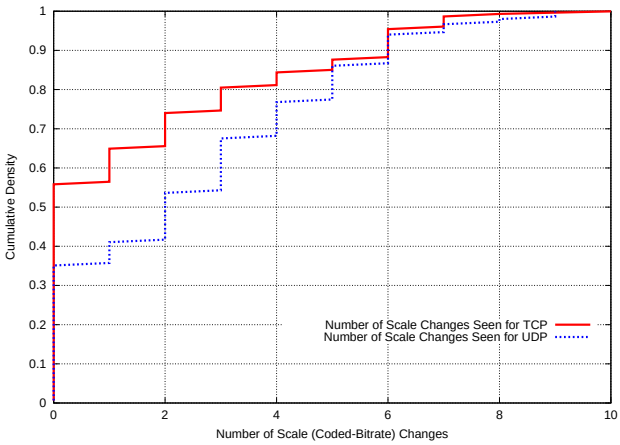

Fig. 9. CDF of Media Scale Changes (DSL: BR=35 Kbps, Q=5 Kbytes)

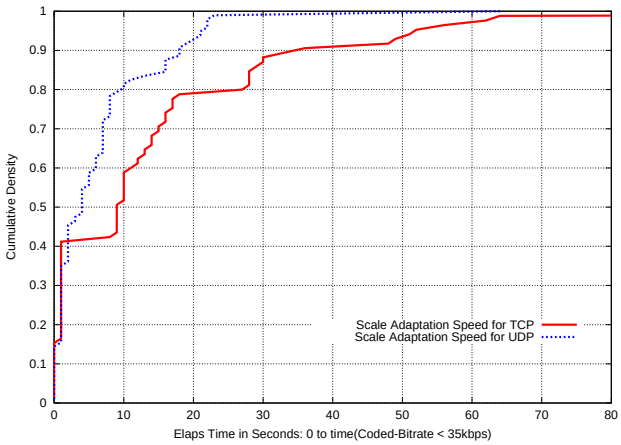

Fig. 10. CDF of Media Scale Adaptation Speed (DSL: BR=35 Kbps, Q=5 Kbytes)

low-capacity connection. Quantitatively, by looking at the end-time of transmission, the top graph of Figure 8 shows that to play 3 minutes (i.e., 180 seconds) 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 a 3 minute video clip, while streaming over TCP required more than 2 minutes of buffering to play the same 3 minute clip.

In Figure 9, the CDFs depict the number of media scale changes seen for each video clip, and summarize the relative responsiveness of RealVideos in scaling the application data rate to below the available network bitrate. Overall, UDP streams had more scale changes than did TCP streams. Also, Figure 9 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 10 summarizes the responsiveness of RealVideo media scaling based on how quickly the video stream adapted to the available bitrate after streaming started. Specifically, for the successfully adapted streams, we measure the time taken for the encoded bitrate to drop under the inbound capacity limit, depicted as the first point under the 35 Kbps limit for each stream in Figure 8. Figure 10 shows that about 15% of the 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 bitrate within a couple of seconds, independently 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 bitrate. For example, 80% of the UDP video streams adapted to the available bitrate 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 bitrate, causing UDP streaming to be unfair under capacity-constrained conditions. However, most RealVideo clips can, and do, scale their application data rates to the available network bitrate. RealVideo streams over UDP can adjust their application data rates to the available bitrate more efficiently than can RealVideo over TCP.

D. *Buffering Data Rate*

As suggested in [16], RealPlayer buffers data at an accelerated rate for the first part of a clip. Confirming and analyzing the rate of this buffering rate versus steady playout rate may help to characterize the bursty nature of RealVideo streams.

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

Figure 11 depicts the ratio of (average buffering data rate / 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. Low bitrate clips buffered at up to 6 times their average playout rates. Higher bitrate clips buffered at relatively lower rates, possibly because capacity restrictions limited them from buffering at a higher rate. The buffering / steady playout ratios less than 1 in the 0-150 Kbps range for some TCP streams are caused by TCP retransmission timeouts during buffering.

In order to determine if capacity restrictions limit buffering rates, we ran a set of experiments with the bottleneck capacity being the campus LAN attached to the Internet via a 15 Mbps link.[14] In this setup, the LAN en-

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

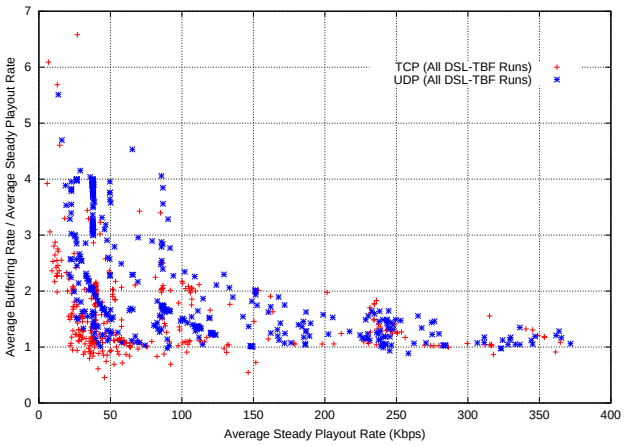

Fig. 11. Ratio of Average Buffering Rate to Average Steady Playout Rate versus Average Steady Playout Rate (all runs)

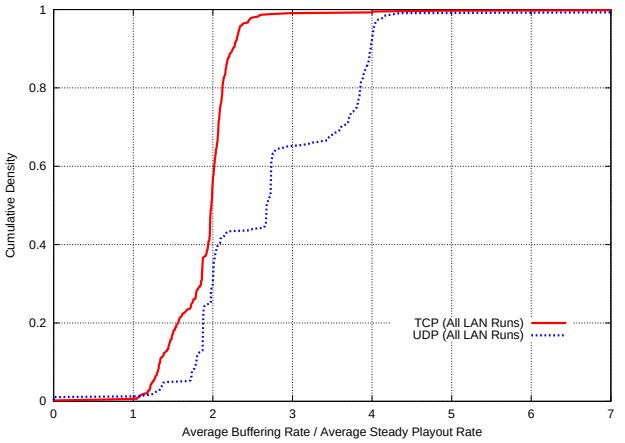

Fig. 12. CDF of Ratio of Average Buffering Rate to Average Steady Playout Rate (LAN)

vironment was relatively unconstrained, having a bottleneck capacity which was typically at least three times that of our 600 Kbps bottleneck capacity.

Figure 12 depicts a CDF of the ratio of the average buffering data rate to the average steady playout rate. The buffering rate to steady rate ratio 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 rate for UDP was significantly higher than that of TCP. For UDP, the vertical "steps" in the CDF are at typical RealVideo 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.

In general, both RealVideo clips over UDP and RealVideo clips over TCP buffer data at a significantly higher rate than the steady playout rate. Due to this prominent buffering period, RealVideo cannot be modeled as a simple CBR flow, as is common in many network simulations that include streaming media. In fact, looking at a simple average bitrate over the length of the entire clip will also not reveal the true nature of RealVideo since it will miss the buffering period. An accurate bitrate distribution for RealVideo must include a buffering stage, whereby the sending data rate is typically from 2-5 times the steady-state playout rate and a post-buffering stage whereby the actual bitrate is dependent on the encoding bitrate of the content and the network conditions.

E. Smoothness

Streaming video requires not only a moderate to high bitrate but also a smooth data rate. TCP's acknowledgment-based window advancement can result in a bursty data rate, thus requiring a significantly larger receiver buffer at the application level for a smooth media playout than is required for UDP streams. Streaming media applications, especially real-time applications, sometimes cite these reasons for choosing UDP as their primary transport protocol [12].

For each clip, we calculate the "smoothness" of the network data rate every 500 ms by taking the ratio of consecutive bitrates measured over each interval. For example, if the data rate is 200 Kbps for one time interval and 400 Kbps the next time interval, the smoothness of the interval would be 2. If the data rate then dropped by half back to 200 Kbps, it would be 0.5 for the next interval. Figure 13 depicts CDFs of smoothness for each network bottleneck bandwidth, with the x-axis drawn in log-scale so as to make a smoothness of 0.5 and 2 visually equal. Both TCP and UDP were smooth for a bottleneck capacity of 600 Kbps. With bottleneck capacities of 300, 150 and 75 Kbps, both TCP and UDP became noticeably less smooth, with TCP often far less smooth than UDP.

In general, as for other streaming applications, streaming RealVideo clips over UDP receive a smoother playout rate than do streaming RealVideo clips over TCP for capacity-constrained conditions.

VI. 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 available bitrate. As seen in Section V, streaming media over UDP can sometimes result in a higher average bitrate than streaming media over TCP, primarily because competing TCP sources are forced to transmit at a reduced rate. Plus, as seen in Section V-C, it is more difficult for the application layer to adjust the encoding rate to the available bitrate when using TCP

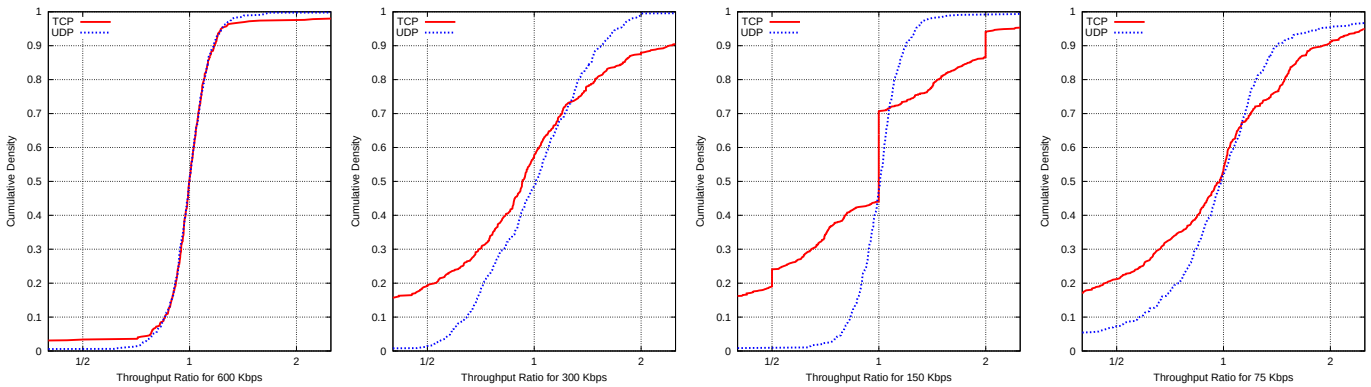

Fig. 13. Smoothness Ratio for Bottleneck Capacities of 600, 300, 150, and 75 Kbps

(because the TCP API does not provide network loss rate, for example). Thus, there are strong application-oriented reasons for streaming media to use UDP rather than TCP, suggesting potentially high-bitrate video over UDP may contribute to congestion collapse.

However, an unresponsive "fire-hose" application, such as high-quality video using UDP 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 [4]. Instead, applications can significantly benefit by using media scaling, as illustrated by RealPlayer in Section V-C, to make intelligent decisions about which packets not to send beforehand, making low quality video over the same congested link quite effective.

As shown in Section V-B, media scaling, a streaming QoS control mechanism, can also be an effective means of responding to network congestion. While scaling the application data rate to meet the available bitrate, RealVideo over UDP often achieves a TCP-Friendly transmission rate. However, these results, obtained in an experimental environment that induces contention at the low-capacity last-mile link, may or may not hold for contention on high-capacity backbone links. Typically, the packet drop rate of a high-capacity link may be affected little by the data rate of a single video stream, causing a weaker control relation between media scaling and network contention (packet loss rate). Under such a condition, the sensitivity of the media scaling will dominate the congestion responsiveness of the UDP video stream. While we were unable to measure the scaling sensitivity of RealVideo streams in this study, in the worst case, a RealVideo may very coarsely react to the network regardless of the scale levels supported by the clip, streaming either at the highest or at the lowest quality level.

In addition, although media scaling, when coupled with properly scale-encoded RealVideo clips, may be effective it is not always guaranteed as media scaling is an optional encoding feature provided to content providers as a means to enhance streaming media quality rather than as a proper congestion control mechanism. Section V-C shows that about 30% of RealVideo streams could not do application media scaling at all, being unresponsive to network congestion when streaming over UDP.

Lastly, the higher buffering rate seen in Section V-D 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 avoids any un-smoothness 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 any competing TCP flows. Overall, from the network point of view, the buffering rate should be limited to the playout rate, and is was for earlier versions of some other commercial players [16].

Thus, despite some positive congestion responsiveness results of RealVideo UDP streams, end-to-end congestion control that relies solely on media scaling may not be a suitable solution for the well-being of the Internet. Instead, a streaming-friendly transport protocol with a proper congestion control mechanism should be provided. Such a protocol would ideally provide a streaming-friendly API that gives applications transmission state information as well as control over the transmission buffer management for efficient media scaling.

VII. Conclusions

In this work, we evaluated 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 allowed us to simultaneously stream two RealVideo clips, one over TCP and

one over UDP, along the same network path. Our testbed also let us control the network bottleneck capacity, thus allowing us to evaluate the responsiveness to congestion of the UDP streams. Using our testbed, we streamed over 600 hours of videos from over 2000 video clips with a variety of content and encoding bandwidths selected from across the Internet.

Overall, we find RealVideo over UDP typically receives bitrates comparable to that of TCP under normal network conditions. Even during periods of packet loss, most RealVideo over UDP is TCP-Friendly. However, under capacity-constrained conditions, RealVideo over UDP can have a higher bitrate than TCP and the bitrate use gets increasingly unfair with an increase in packet loss rate and round-trip time.

Media scaling directly determines the congestion responsiveness of UDP streams and can be an effective means of responding to congestion when paired with properly scale-encoded video clips. However, properly scale-encoded video clips are not guaranteed as they are an optional encoding feature provided as a means to enhance streaming media quality rather than as a proper congestion control mechanism. In addition, the user-beneficial initial burst of buffering traffic over UDP, much higher than the average playout rate, can cause considerable congestion and can make RealVideo network traffic more difficult to manage.

This study concludes that while not threatening the well-being of the Internet as is commonly feared under normal network conditions, RealVideo UDP streams may also not necessarily be good Internet citizens. Furthermore, since our observations apply only to RealVideo streams with the congestion responsiveness of other popular streaming applications still unknown, we see the need for a streaming friendly network protocol that supports end-to-end congestion control.

We also analyzed the suitability and/or shortcomings of using TCP as a streaming transport protocol. Adjusting the application data rate to the available network bitrate 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. This suggests that streaming friendly application programming interface (API) design is one of the most important factors that will aid a successful deployment of a future streaming transport protocol.

VIII. 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 on the Internet might help understand the differences in congestion responsiveness across commercial players. We have conducted preliminary comparisons of RealPlayer and Media Player in [16] and measured the responsiveness of Media Player in a controlled, non-Internet environment in [22].

In this study, 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 [32]. 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.

The work in this paper did not explore the relationship between perceptual quality of the video, influenced by application level performance 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 the available bitrate.

IX. Acknowledgments

We would like to acknowledge Yali Zhu for her help in the initial collection of the data. We would also like to thank the anonymous reviewers for their detailed comments on an earlier version of this paper.

References

[1] A. Akella, S. Seshan, and A. Shaikh, "An Empirical Evaluation of Wide-Area Internet Bottlenecks," in *Proceedings of the ACM Internet Measurement Conference (IMC)*, Oct. 2003. [Online]. Available: http://www.icir.org/vern/imc-2003/papers/p303-akella.pdf

[2] 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.

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

[4] J. Boyce and R. Gaglianello, "Packet Loss Effects on MPEG Video sent over the Public Internet," in *Proceedings of ACM Multimedia*, Bristol, U.K., Sept. 1998, pp. 181–190.

[5] Z. Cao, Z. Wang, and E. Zegura, "Rainbow Fair Queuing: Fair Bandwidth Sharing Without Per-Flow State," in *Proceedings of IEEE Infocom*, Tel-Aviv, Israel, Mar. 2000, pp. 922 – 931.

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

[6] 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)*, San Francisco, CA, USA, Mar. 2001, pp. 1–12.

[7] 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.

[8] 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.

[9] W. Feng, D. Kandlur, D. Saha, and K. Shin, "Stochastic Fair Blue: A Queue Management Algorithm for Enforcing Fairness," in *Proceedings of IEEE Infocom*, Anchorage, Alaska, USA, Apr. 2001, pp. 1520 – 1529.

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

[11] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-Based Congestion Control for Unicast Applications," in *Proceedings of ACM SIGCOMM Conference*, Stockholm, Sweden, Aug. 2000, pp. 43 – 56.

[12] A. Goel, C. Krasic, K. Li, and J. Walpole, "Supporting Low Latency TCP-Based Media Streams," in *Proceedings of International Workshop on Quality of Service (IWQoS)*, Miami Beach, FL, USA, May 2002.

[13] 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.

[14] T. Kuang and C. Williamson, "A Measurement Study of RealMedia Audio/Video Streaming Traffic," in *Proceedings of ITCOM*, Boston, MA, USA, July 2002, pp. 68–79.

[15] K. Lakshminarayanan and V. Padmanabhan, "Some Findings on the Network Performance of Broadband Hosts," in *Proceedings of the Internet Measurement Conference (IMC)*, Miami, FL, USA, October 2003.

[16] M. Li, M. Claypool, and R. Kinicki, "MediaPlayer versus RealPlayer – A Comparison of Network Turbulence," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop (IMW)*, Marseille, France, Nov. 2002, pp. 131 – 136.

[17] D. Lin and R. Morris, "Dynamics of Random Early Detection," in *Proceedings of ACM SIGCOMM Conference*, Sept. 1997.

[18] 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.

[19] R. Mahajan, S. Floyd, and D. Wetherall, "Controlling High-Bandwidth Flows at the Congested Routers," in *Proceedings of the 9th International Conference on Network Protocols (ICNP)*, Mission Inn, Riverside, CA, USA, Nov. 2001, pp. 192 – 201.

[20] A. Mena and J. Heidemann, "An Empirical Study of Real Audio Traffic," in *Proceedings of IEEE Infocom*, Tel-Aviv, Israel, Mar. 2000, pp. 101 – 110.

[21] 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.

[22] J. Nichols, M. Claypool, R. Kinicki, and M. Li, "Measurements of the Congestion Responsiveness of Windows Streaming Media," in *Proceedings of the 14th ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2004.

[23] C. Padhye, K. Christensen, and W. Moreno, "A New Adaptive FEC Loss Control Algorithm for Voice Over IP Applications," in *Proceedings of IEEE International Performance, Computing and Communication Conference*, Feb. 2000.

[24] 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.

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

[26] Real Networks Incorporated, "RealPlayer 8 User Manual," copyright 2000, URL: http://service.real.com/help/player-/plus_manual.8/rppmanual.htm.

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

[28] R. Rejaie, M. Handley, and D. Estrin, "RAP: An End-to-end Rate-based Congestion Control Mechanism for Realtime Streams in the Internet," in *Proceedings of IEEE Infocom*, New York, NY, Mar. 1999, pp. 1337 – 1345.

[29] 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*, Vancouver, British Columbia, Canada, Sept. 1998, pp. 118 – 130.

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

[31] J. van der Merwe, R. Caceres, Y. hua Chu, and C. Sreenan, "mmdump - A Tool for Monitoring Internet Multimedia Traffic," *ACM Computer Communication Review*, vol. 30, no. 5, pp. 48–59, Oct. 2000.

[32] E. Veloso, V. Almeida, W. Meira, A. Bestavros, and S. Jin, "A Hierarchical Characterization of a Live Streaming Media Workload," in *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, Marseille, France, Nov. 2002, pp. 117 – 130.

[33] 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.

[34] 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 (IMW)*, San Francisco, California, USA, Nov. 2001, pp. 295 – 309.