

End-to-End Internet Video Traffic Dynamics: Statistical Study and Analysis

Dmitri Loguinov
Computer Science Department
City University of New York
New York, NY 10016
csdsl@cs.ccny.cuny.edu

Hayder Radha
Dept. of Electrical & Computer Engineering
Michigan State University
East Lansing, MI 48824
radha@egr.msu.edu

Abstract – In this paper, we analyze the dynamics of a seven-month real-time streaming experiment, which was conducted between a number of unicast dialup clients, connecting to the Internet through access points in more than 600 major U.S. cities, and a backbone video server. During the experiment, the clients streamed low-bitrate MPEG-4 video sequences from the server over paths with more than 5,000 distinct Internet routers. We describe the methodology of the experiment and the architecture of our NACK-based streaming application, study end-to-end dynamics of 16 thousand ten-minute sessions (85 million packets), and analyze the behavior of the following network parameters: packet loss, round-trip delay, and packet reordering. We also study the impact of these parameters on the quality of real-time streaming (i.e., the number of underflow events).

I. INTRODUCTION

The behavior of the Internet has been the target of numerous studies, but nevertheless, its performance from the perspective of an average home user still remains relatively undocumented. At the same time, we believe that since end users are responsible for a large fraction of Internet traffic, the study of network conditions experienced by these users is an important research topic. This is the reason that compelled us to conduct a fundamentally different performance study that looks at Internet dynamics from the angle of an average Internet user.

In addition, even though the end-to-end performance of the Internet has been extensively analyzed in the past, an overwhelming majority of previous studies were based on TCP or ICMP traffic. On the other hand, real-time streaming protocols have not received as much attention in these studies. In fact, the dynamics of NACK-based UDP protocols (not necessarily real-time) have been the focus of only a few research studies (e.g., [14]).

The Internet has been studied using TCP traffic by Paxson [18], Bolliger *et al.* [3], Caceres *et al.* [7], Mogul [16], and several others (e.g., [2]). Paxson's study included 35 geographically distributed sites in 9 countries; Bolliger *et al.* employed 11 sites in 7 countries; whereas the majority of other researchers monitored transit TCP traffic at a single backbone router [2], [16] or inside several campus networks [7] for the duration ranging from several hours to several days. In addition, several other studies employed ICMP probes [1], [17], [18], UDP echo packets [5], [6], and multicast backbone (MBone) audio packets [25]. With the exception of the last study, neither the setup, nor the type of probe traffic of prior work represented realistic real-time streaming scenarios.

Among the studies that specifically sent audio/video traffic over the Internet (e.g., [4], [8], [22], [23]), the majority of experiments involved only a few Internet paths, lasted for a short period of time, and focused on analyzing the features of the

proposed scheme rather than the impact of Internet conditions on real-time streaming.

The methodology used in prior large-scale TCP experiments (e.g., [3], [18]) was similar and involved a topology where each participating site was paired with every other participating site for an FTP-like transfer. Although this setup approximates well the current use of TCP in the Internet, future entertainment-oriented streaming services, however, are more likely to involve a small number of backbone video servers and a large number of home users.¹

We believe that in order to study the current dynamics of real-time streaming in the Internet, we must take the same steps to connect to the Internet as an average end-user² (i.e., through dialup ISPs). For example, ISPs often experience congestion in their own backbones, and during busy hours, V.90 modems in certain access points are not available due to high user demand, none of which can be captured by studying the Internet from a small campus network directly connected to the Internet backbone.

Furthermore, the sending rate of a TCP connection is driven by its congestion control, which can often cause increased packet loss and higher end-to-end delays in the path along which it operates (e.g., during slow start). In our experiment, we measured *true* end-to-end path dynamics without the bias of congestion control applied to slow modem links.³ Additionally, our decision not to use congestion control was influenced by the evidence that the majority of streaming traffic in the current Internet employs constant-bitrate (CBR) video streams [20], where the user explicitly selects the desired streaming rate from the corresponding web page (note that the additional rate adaptation implemented in [20] is very rudimentary and could hardly be considered congestion control).

In this paper, we present the methodology and analyze the results of a seven-month large-scale real-time streaming experiment, which involved three nation-wide dialup ISPs, each with several million active subscribers in the United States. The topology of the experiment consisted of a backbone video server streaming MPEG-4 video sequences to unicast home users located in more than 600 major U.S. cities. The streaming was performed in real-time (i.e., with a real-time decoder), utilized UDP for the transport of all messages, and relied on

¹ Our work focuses on non-interactive streaming applications where the user can tolerate short (i.e., in the order of several seconds) startup delays (e.g., TV over the Internet).

² Recent market research reports (e.g., [9], [11]) show that in Q2 of 2001, approximately 87-89% of U.S. households used dialup access to connect to the Internet. Furthermore, it is predicted [9], [10] that even in 2005, the majority of U.S. households will still be using dialup modems.

³ Without a doubt, future real-time streaming protocols will include some form of scalable congestion control.

simple NACK-based retransmission to recover lost packets before their decoding deadlines.

Even though we consider it novel and unique in many aspects, there is an important limitation to our study. Our experiments document Internet path dynamics perceived by *low-bitrate* streaming sessions. Recall that one of the goals of our work was to conduct a performance study of the Internet from the angle of a typical home Internet user, and to this extent, we consider our work to be both thorough and successful. In addition, by focusing on low-bitrate paths, our study shows the performance of real-time protocols under the most difficult network conditions (i.e., large end-to-end delays, relatively high bit-error rates, low available bandwidth, etc.) and provides a “lower bound” on the performance of future Internet streaming applications.

The remainder of the paper is organized as follows. Section II describes the methodology of the experiment. Section III provides an overview of the experiment. Section IV studies packet loss, section V analyzes the frequency of underflow events, section VI looks at the behavior of the round-trip delay, and section VII examines the extent of packet reordering during the experiment. Section VIII concludes the paper.

II. METHODOLOGY

A. Setup of the Experiment

We started our work by attaching a Unix video server (located at Philips Research USA) to the UUNET backbone via a T1 link (Figure 1). To support the client’s connectivity to the Internet, we selected three major nation-wide dialup ISPs (which we call ISP_a , ISP_b , and ISP_c), each with at least five hundred V.90 (i.e., 56 kb/s) dialup numbers in the U.S., and designed an experiment in which hypothetical Internet users dialed a local access number to reach the Internet and streamed video sequences from the server. Although the clients were physically placed in our lab in the state of New York, they dialed long-distance phone numbers and connected to the Internet through ISPs’ access points located in each of the 50 U.S. states. Our database of phone numbers included 1,813 different V.90 access points in 1,188 major U.S. cities.

After the phone database was in place, we designed and implemented special software, which we call the *dialer*, that dialed phone numbers from the database, connected to the ISPs using the point-to-point protocol (PPP), and issued a *parallel traceroute*⁴ to the server. Upon success of the traceroute, the dialer started the video client with the instructions to stream a ten-minute video sequence from the server.

In our analysis of the data, we attempted to isolate clearly modem-related pathologies (such as packet loss caused by a poor connection over the modem link and large RTTs due to data-link retransmission) from those caused by congested routers of the Internet. Thus, connections that were unable to complete a traceroute to the server, connections with high bit-error rates (BER), and connections during which the modem could not sustain our streaming rates were all considered useless for our study and were excluded from the analysis in this paper.

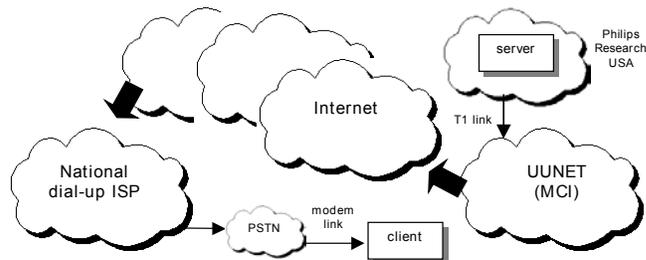


Figure 1. Topology of the experiment.

In order to make the experiment reasonably short, we considered all phone numbers from the same state to be equivalent, and consequently, we assumed that a successful streaming attempt through any phone number of a state indicated a *successful coverage* of the state regardless of which phone number was used. Furthermore, we divided each 7-day week into 56 three-hour timeslots (i.e., 8 timeslots per day) and designed the dialer to select phone numbers from the database in such order so that each state would be successfully covered within each of the 56 timeslots at least once. In other words, each ISP needed to sustain exactly $50 \cdot 56 = 2,800$ successful sessions before the experiment was allowed to end.

B. Real-time Streaming

For the purpose of the experiment, we used an MPEG-4 encoder to create two ten-minute QCIF (176x144) video streams. The first stream, which we call S_1 , was coded at the video bitrate of 14 kb/s (size 1.05 MBytes), and the second stream, which we call S_2 , was coded at 25 kb/s (size 1.87 MBytes). The experiment with stream S_1 lasted during November – December 1999 and the one with stream S_2 was an immediate follow-up during January – May 2000.

During the transmission of each video stream, the server split it into 576-byte IP packets. Stream S_1 consisted of 4,188 packets, and stream S_2 consisted of 5,016 packets. Video frames always started on a packet boundary, and consequently, the last packet in each frame was allowed to be smaller than others (in fact, many P (prediction-coded) frames were smaller than the maximum payload size and were carried in a single UDP packet). As a consequence of the packetization overhead, the IP bitrates (i.e., including IP, UDP, and our special 8-byte headers) for streams S_1 and S_2 were 16.0 and 27.4 kb/s, respectively.

In our streaming experiment, the term *real-time* refers to the fact that the video decoder was running in real-time. Recall that each compressed video frame has a specific *decoding deadline*, which is usually based on the time of the frame’s encoding. If a compressed video frame is not fully received by the decoder buffer at the time of its deadline, the video frame is discarded and an underflow event is registered.

In order to compensate for one-way delay jitter and allow retransmissions, many real-time streaming protocols *pre-buffer* video data before starting the decoding process [8], [20]. The duration of such pre-buffering is called the *startup delay*⁵ of a

⁴ The ICMP traceroute probes were sent in parallel instead of sequentially to facilitate a quicker discovery of intermediate routers.

⁵ Note that many video coding schemes require that an *additional* startup delay be applied to the decoder buffer. This additional delay is called the *ideal startup delay* and is derived from the encoder buffer model [19], [21]. To avoid confusion, we entirely avoid the discussion of the ideal startup delay in this paper.

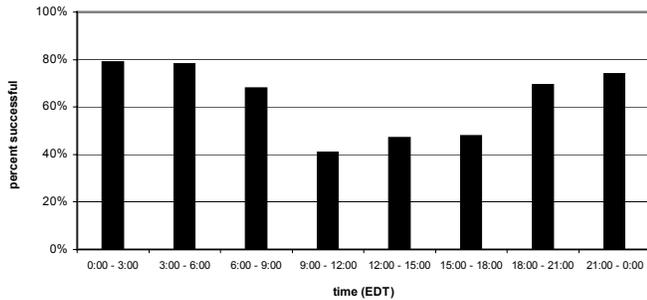


Figure 2. Success of streaming attempts during the day.

session. In all our experiments, we used a startup delay equal to 2,700 ms, which was manually selected based on preliminary testing.

C. Client-Server Architecture

For the purpose of our experiment, we implemented a client-server architecture for MPEG-4 streaming over the Internet. The server was fully multithreaded to ensure that the transmission of packetized video was performed at the target IP bitrate of each streaming session and to provide quick response to clients' NACK requests. The streaming was implemented in bursts of packets (with the burst duration D_b varying between 340 and 500 ms depending on the bitrate) for the purposes of making the server as low-overhead as possible (for example, RealAudio servers use $D_b = 1,800$ ms [15]). Although we agree that in many cases the desired way of sending constant bitrate (CBR) traffic is to equally space packets during transmission, there are practical limitations (such as OS scheduling and inter-process switching delays) that often do not allow us to follow this model.

The second and the more involved part of our architecture, the client, was designed to recover lost packets through NACK-based retransmission and collect extensive statistics about each received packet and each decoded frame. Furthermore, as it is often done in NACK-based protocols, the client was in charge of collecting round-trip delay (RTT) samples.⁶ The measurement of the RTT involved the following two methods. In the first method, each successfully recovered packet provided a sample of the RTT (i.e., the RTT was the duration between sending a NACK and receiving the corresponding retransmission). In our experiment, in order to avoid the ambiguity of which retransmission of the same packet actually returned to the client, the header of each NACK request and each retransmitted packet contained an extra field specifying the retransmission attempt of the packet.

The second method of measuring the RTT was used by the client to obtain *additional* samples of the round-trip delay in cases when network packet loss was too low. The method involved periodically sending *simulated* retransmission requests to the server if packet loss was below a certain threshold. In response to these simulated NACKs, the server included the usual overhead⁷ of fetching the needed packets from the storage and sending them to the client. In our experiment, the client activated simulated NACKs, spaced 30 seconds apart, if packet loss was below 1%.

⁶ The resolution of the timestamps was 100 microseconds.

⁷ The server overhead was below 10 ms for all retransmitted packets.

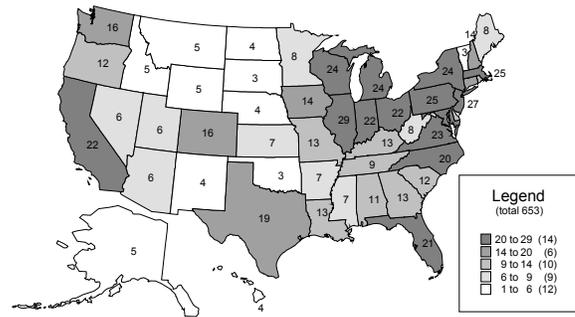


Figure 3. The number of cities per state that participated in either D_1 or D_2 .

We tested the software and the concept of a wide-scale experiment of this sort for nine months before we felt comfortable with the setup, the reliability of the software, and the exhaustiveness of the collected statistics. In addition to extensive testing of the prototype for nine months, we monitored various statistics reported by the clients in real-time (i.e., on the screen) during the experiments for sanity and consistency with previous tests.

Our traces consist of six datasets, each collected by a different machine. Throughout this paper, we will use notation D_n^x to refer to the dataset collected by the client assigned to ISP $_x$ ($x = a, b, c$) during the experiment with stream S_n ($n = 1, 2$). Furthermore, we will use notation D_n to refer to the combined set $\{D_n^a \cup D_n^b \cup D_n^c\}$.

III. EXPERIMENT OVERVIEW

In dataset D_1 , the three clients performed 16,783 long-distance connections to the ISPs' remote modems and successfully completed 8,429 streaming sessions.⁸ In D_2 , the clients performed 17,465 modem connections and sustained 8,423 successful sessions. Analysis of the above numbers suggests that in order to receive real-time streaming material with a minimum quality at 16 to 27.4 kb/s, an average U.S. end-user, equipped with a V.90 modem, needs to make approximately two dialing attempts to the ISPs' phone numbers within the state where the user resides. The success rate of streaming sessions during different times of the day is illustrated in Figure 2. Note the dip by a factor of two between the best and the worst times of the day.

Furthermore, in dataset D_1 , the clients traced the arrival of 37.7 million packets, and in D_2 , the arrival of additional 47.3 million (for a total of 85 million). In terms of bytes, the first experiment transported 9.4 GBytes of video data and the second one transported another 17.7 GBytes (for a total of 27.1 GBytes).

Recall that each experiment lasted for as long as it was necessary to cover the entire United States. Depending on the success rate within each state, the access points used in the experiment comprised a subset of our database. In D_1 , the experiment covered 962 dialup points in 637 U.S. cities, and in D_2 , it covered 880 dialup points in 575 U.S. cities. Figure 3 shows the combined (i.e., including both datasets D_1 and D_2) number of distinct cities in each state covered by our experiment (1,003 access points in 653 cities).

⁸ Typical reasons for failing a session were PPP-layer connection problems, inability to reach the server (i.e., failed traceroute), high bit-error rates, low (14.4-19.2 kb/s) connection rates, and insufficient bandwidth.

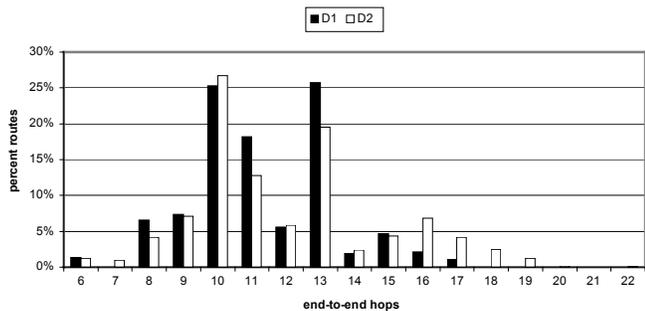


Figure 4. Distribution of the number of end-to-end hops.

During the experiment, each session was preceded by a parallel traceroute, which recorded the IP addresses of all discovered routers (DNS and WHOIS⁹ lookups were done off-line after the experiments were over). Dataset D_1 recorded 3,822 distinct Internet routers, D_2 recorded 4,449 distinct routers, and both experiments combined produced the IP addresses of 5,266 unique routers. The majority of the discovered routers belonged to the ISPs' networks (51%) and UUNET (45%), which confirmed our intuition that all three ISPs had direct peering connections with UUNET. Moreover, our traces recorded approximately 200 routers that belonged to five additional Autonomous Systems (AS).

The average end-to-end hop count was 11.3 in D_1 (6 minimum and 17 maximum) and 11.9 in D_2 (6 minimum and 22 maximum). Figure 4 shows the distribution of the number of hops in the encountered end-to-end paths in each of D_1 and D_2 . As the figure shows, the majority of paths (75% in D_1 and 65% in D_2) contained between 10 and 13 hops.

Throughout the rest of the paper, we restrict ourselves to studying only *successful* (as defined in section II.A) sessions in both D_1 and D_2 . We call these new purged datasets (with only successful sessions) D_{1p} and D_{2p} , respectively (purged datasets D_{np}^x are defined similarly for $n = 1, 2$ and $x = a, b, c$). Recall that $\{D_{1p} \cup D_{2p}\}$ contains 16,852 successful sessions, which are responsible for 90% of the bytes and packets, 73% of the routers, and 74% of the U.S. cities recorded in $\{D_1 \cup D_2\}$.

IV. PACKET LOSS

A. Overview

Numerous researchers have studied Internet packet loss, and due to the enormous diversity of the Internet, only few studies agree on the average packet loss rate and the average *loss burst length* (i.e., the number of packets lost in a row). Among numerous studies, the average Internet packet loss was reported to vary between 11% and 23% by Bolot [5] depending on the inter-transmission spacing between packets, between 0.36% and 3.54% by Borella *et al.* [6] depending on the studied path, between 1.38% and 11% by Yajnik *et al.* [25] depending on the location of the Mbone receiver, and between 2.7% and 5.2% by Paxson [18] depending on the year of the experiment. In addition, 0.49% average packet loss rate was recently reported by Balakrishnan *et al.* [2], who analyzed the dynamics of a large number of TCP web sessions at a busy Internet server.

⁹ The WHOIS database was used to discover the Autonomous System (AS) of each router.

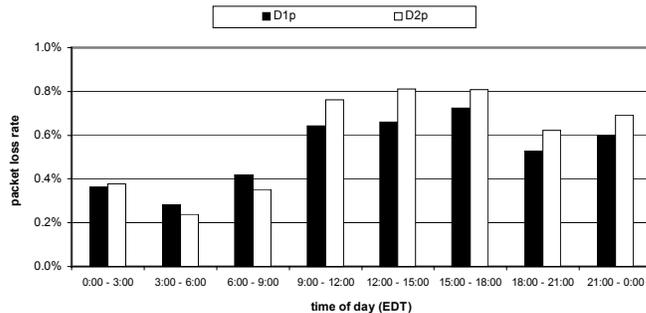


Figure 5. Average packet loss rates during the day.

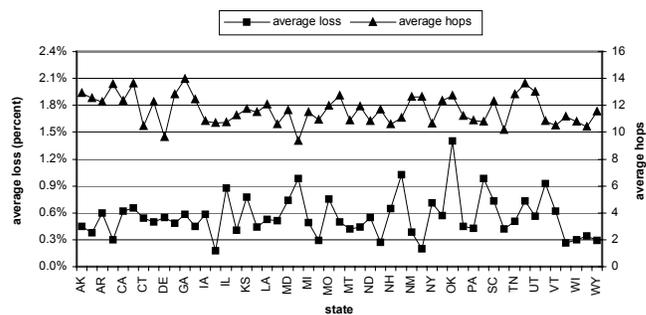


Figure 6. Average per-state packet loss rates.

In dataset D_{1p} , the average recorded packet loss rate was 0.53% and in D_{2p} , it was 0.58%. Even though these rates are much lower¹⁰ than those traditionally reported by Internet researchers during the last decade, they are still somewhat higher than those reported by backbone ISPs [24]. Furthermore, 38% of the sessions in $\{D_{1p} \cup D_{2p}\}$ did not experience any packet loss, 75% experienced loss rates below 0.3%, and 91% experienced loss rates below 2%. On the other hand, 2% of the sessions suffered packet loss rates 6% or higher.

In addition, as we expected, average packet loss rates exhibited a wide variation during the day. Figure 5 shows the evolution of loss rates as a function of the timeslot (i.e., the time of day), where each point represents the average of approximately 1,000 sessions. As the figure shows, the variation in loss rates between the best (3-6 am) and the worst (3-6 pm) times of the day was almost by a factor of four.

The apparent discontinuity in Figure 5 between timeslots 7 (21:00-0:00) and 0 (0:00-3:00) is due to the coarse timescale used in the figure. On finer timescales (e.g., minutes), loss rates converge to a common value near midnight. A similar discontinuity in packet loss rates was reported by Paxson [18] for North American sites, where packet loss during timeslot 7 was approximately twice as high as that during timeslot 0.

The variation in the average *per-state* packet loss (as shown in Figure 6) was quite substantial (from 0.2% in Idaho to 1.4% in Oklahoma), but virtually did not depend on the state's average number of end-to-end hops (correlation coefficient ρ was only -0.04) or the state's average RTT (correlation -0.16). However, as we will see later, the average per-state RTT and

¹⁰ Note that during the experiment, simply dialing a different access number in most cases fixed the problem of high packet loss. This fact shows that the majority of failed sessions documented pathologies created by the modem (or the access point) rather than the actual packet loss in the Internet. Since an end-user can always re-dial a bad connection searching for better network conditions, we believe that the bias created by removing failed sessions reflects the actions of a typical Internet user.

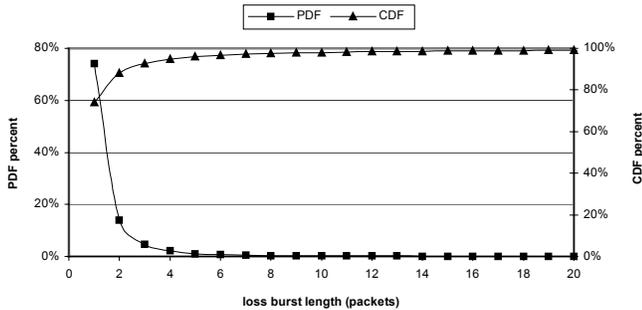


Figure 7. PDF and CDF functions of loss burst lengths in $\{D_{1p} \cup D_{2p}\}$.

the number of end-to-end hops were in fact positively correlated.

B. Loss Burst Lengths

We next attempt to answer the question of how bursty Internet packet loss was during the experiment. Figure 7 shows the distribution (both the PDF and the CDF) of loss burst lengths in $\{D_{1p} \cup D_{2p}\}$ (without loss of generality, the figure stops at burst length 20, covering more than 99% of the bursts). Even though the upper tail of the distribution had very few samples, it was fairly long and reached burst lengths of over 100 packets.

Figure 7 is based on 207,384 loss bursts and 431,501 lost packets. The prevalence of single packet losses, given the fact that packets in our experiment were injected into the Internet in bursts at the T1 speed¹¹, leads us to speculate that either router queues sampled in our experiment overflowed on timescales smaller than the time needed to transmit a single IP packet over a T1 link (i.e., 3 ms for the largest packets and 1.3 ms for the average-size packets), or that backbone routers employed Random Early Detection (RED) for preventing congestion. However, a second look at the situation reveals that server’s possible interleaving of packets from different sessions could have expanded each flow’s inter-packet transmission distance. Furthermore, before each lost packet reached the corresponding congested router, packets from other Internet flows could have queued immediately behind the lost packet, effectively expanding the inter-packet distance even further. Therefore, our initial speculation about the duration of buffer overflows during the experiment may not hold in all cases.

On the other hand, to investigate the presence of RED in the Internet, we contacted several backbone and dialup ISPs whose routers were recorded in our trace data and asked them to comment on the deployment of RED in their backbones. Among the ISPs that responded to our request, the majority had purposely disabled RED and the rest were running RED only for select customers at border routers, but not on the public backbone. Consequently, we conclude that even though the analysis of our datasets points towards transient (i.e., 1-3 ms) buffer overflows in the Internet routers sampled by our experiment, we cannot reliably determine the exact duration of these events.

As previously pointed out by many researchers [25], the upper tail of loss burst lengths usually contains a substantial per-

¹¹ The server was only involved in low-bitrate streaming for our clients and did not have a problem blasting bursts of packets at the full speed of the adjacent link (i.e., 10 mb/s). The spacing between packets was further expanded by the T1 link to UUNET.

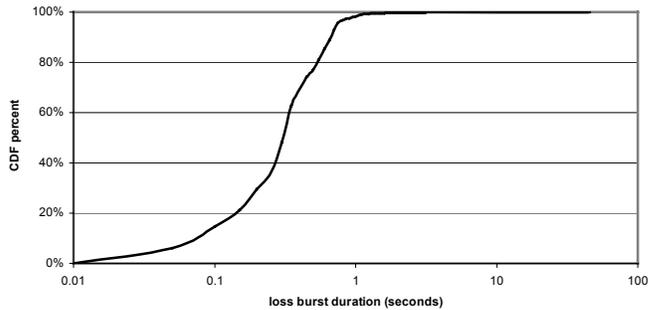


Figure 8. The CDF function of loss burst durations in $\{D_{1p} \cup D_{2p}\}$.

centage of all lost packets. In each of D_{1p} and D_{2p} , single-packet bursts contained only 36% of all lost packets, bursts two packets or shorter contained 49%, bursts 10 packets or shorter contained 68%, and bursts 30 packets or shorter contained 82%. At the same time, 13% of all lost packets were dropped in bursts at least 50 packets long.

Traditionally, the burstiness of packet loss is measured by the average loss burst length. In the first dataset (D_{1p}), the average burst length was 2.04 packets. In the second dataset (D_{2p}), the average burst length was slightly higher (2.10), but not high enough to conclude that the higher bitrate of stream S_2 was clearly responsible for burstier packet loss. Furthermore, the *conditional probability* of packet loss, given that the previous packet was also lost, was 51% in D_{1p} and 53% in D_{2p} . These numbers are consistent with those previously reported in the literature. Bolot [5] observed the conditional probability of packet loss to range from 18% to 60% depending on inter-packet spacing during transmission, Borella *et al.* [6] from 10% to 35% depending on the time of day, and Paxson [18] reported 50% conditional probability for *loaded* (i.e., queued behind the previous) TCP packets and 25% for *unloaded* packets. Using Paxson’s terminology, the majority of our packets were *loaded* since the server sent packets in bursts at a rate higher than the bottleneck link’s capacity.

C. Loss Burst Durations

To a large degree, the average loss burst length depends on how closely the packets are spaced during transmission. Assuming that bursty packet loss comes from buffer overflow events in drop-tail queues rather than from consecutive hits by RED or from bit-level corruption, it is clear that all packets of a flow passing through an overflowed router queue will be dropped for the duration of the instantaneous congestion. Hence, the closer together the flow’s packets arrive to the router, the more packets will be dropped during each queue overflow. This fact was clearly demonstrated in Bolot’s experiments [5], where UDP packets spaced 8 ms apart suffered larger loss burst lengths (mean 2.5 packets) than packets spaced 500 ms apart (mean 1.1 packets). Yajnik *et al.* [25] reported a similar correlation between loss burst lengths and the distance between packets. Consequently, instead of analyzing burst lengths, one might consider analyzing burst durations since the latter does not depend on inter-packet spacing during transmission.

Using our traces, we can only infer an approximate duration of each loss burst, because we do not know the *exact* time when the lost packets were supposed to arrive to the client (the exact arrival time depends on the end-to-end delay jitter suf-

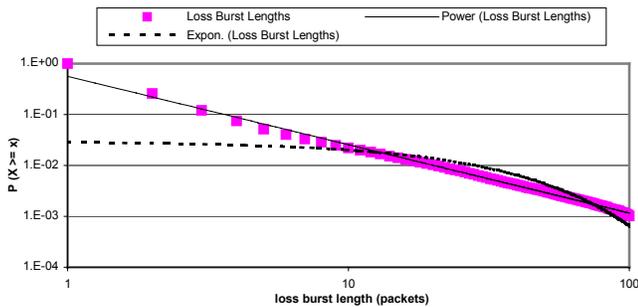


Figure 9. The complimentary CDF of loss burst lengths in $\{D_{1p} \cup D_{2p}\}$ on a log-log scale fitted with hyperbolic (straight line) and exponential (dotted curve) distributions.

ferred by each lost packet). Hence, for each loss event, we define the *loss burst duration* as the time elapsed between the receipt of the packet immediately preceding the loss burst and the packet immediately following the loss burst. Figure 8 shows the distribution (CDF) of loss burst durations in seconds. Although the distribution tail is quite long (up to 36 seconds), the majority (more than 98%) of loss burst durations in both datasets D_{1p} and D_{2p} fall under 1 second. Paxson’s study [18] also observed large loss burst durations (up to 50 seconds), however, only 60% of the loss bursts studied by Paxson were contained below 1 second. In addition, our traces showed that the average distance between lost packets in the experiment was 172-188 good packets, or 21-27 seconds, depending on the streaming rate.

D. Heavy Tails

In conclusion of this section, it is important to note that packet losses sometimes cannot be modeled as independent events due to buffer overflows that last long enough to affect multiple adjacent packets. Consequently, future real-time protocols should expect to deal with bursty packet losses (Figure 7) and possibly heavy-tailed distributions of loss burst lengths (see below).

Several researchers reported a heavy-tailed nature of loss burst lengths, and the shape parameter α of the Pareto distribution fitted to the length (or duration) of loss bursts was recorded to range from 1.06 (Paxson [18]) to 2.75 (Borella *et al.* [6]). On the other hand, Yajnik *et al.* [25] partitioned the collected data into stationary segments and reported that loss burst lengths could be modeled as exponential (i.e., not heavy-tailed) within each stationary segment. In addition, Zhang *et al.* [26] reported that packet loss along some Internet paths was stationary and could be modeled as exponential, whereas other paths were found to be non-stationary and not easy to model.

Using intuition, it is clear that packet loss and RTT random processes in both D_{1p} and D_{2p} are expected to be non-stationary. For example, the non-stationarity can be attributed to the time of day or the location of the client. In either case, we see three approaches to modeling such non-stationary data. In the first approach, we would have to analyze 16,852 PDF functions (one for each session) for stationarity and heavy tails. Unfortunately, an average session contained only 24 loss bursts, which is insufficient to build a good distribution function for a statistical analysis.

The second approach would be to combine all sessions into groups, which are intuitively perceived to be stationary (e.g., according to the access point or the timeslot), and then perform

similar tests for stationarity and heavy tails within each group. We might consider this direction for future work.

The third approach is to do what the majority has done in the past – assume that all data samples belong to a stationary process and are drawn from a single distribution. Using this last approach, Figure 9 shows a log-log plot of the complementary CDF function from Figure 7 with a least-squares fit of a straight line representing a hyperbolic (i.e., heavy-tailed) distribution (the dotted curve is the exponential distribution fitted to the data). The fit of a straight line is excellent (with correlation $\rho = 0.99$) and provides a strong indication that the distribution of loss burst lengths in the combined dataset $\{D_{1p} \cup D_{2p}\}$ is heavy-tailed. Furthermore, as expected, we notice that the exponential distribution in Figure 9 decays too quickly to even remotely fit the data.

Finally, consider a Pareto distribution with CDF $F(x) = 1 - (\beta/x)^\alpha$ and PDF $f(x) = \alpha\beta^\alpha x^{-\alpha-1}$, where α is the shape parameter and β is the location parameter. Using Figure 9, we establish that a Pareto distribution with $\alpha = 1.34$ (finite mean, but infinite variance) and $\beta = 0.65$ fits our data very well.

V. UNDERFLOW EVENTS

The impact of packet losses on real-time applications is understood fairly well. Each lost packet that is not recovered before its deadline causes an underflow event. In addition to packet loss, real-time applications suffer from large end-to-end delays. However, not all types of delay are equally important to real-time applications. As we will show below, one-way delay jitter was responsible for 90 times more underflow events in our experiment than packet loss combined with large RTTs.

Delays are important for two reasons. First, large round-trip delays make retransmissions late for their decoding deadlines. However, the RTT is important only to the extent of recovering lost packets and, in the worst case, can cause only *lost* packets to be late for decoding. On the other hand, the second kind of delay, delay jitter (i.e., one-way delay variation), can potentially cause each *data* (i.e., non-retransmitted) packet to be late for decoding.

Consider the following. In $\{D_{1p} \cup D_{2p}\}$, packet loss affected 431,501 packets, out of which 159,713 (37%) were discovered to be missing *after* their decoding deadlines had passed, and consequently, NACKs were not sent for these packets. Out of 271,788 remaining lost packets, 257,065 (94.6%) were recovered before their deadlines, 9,013 (3.3%) arrived late, and 5,710 (2.1%) were never recovered (even after multiple retransmission attempts). The fact that more than 94% of “recoverable” lost packets were actually received before their deadlines indicates that retransmission is a very efficient method of overcoming packet loss in non-interactive real-time applications. Clearly, the success rate will be even higher in networks with smaller end-to-end delays.

Before we study underflow events caused by delay jitter, let us introduce two types of *late* retransmissions. The first type consists of packets that arrived after the decoding deadline of the last frame of the corresponding group of pictures (GOP). These packets were completely useless and were discarded. The second type of late packets, which we call *partially late*, consists of those packets that missed their own decoding deadline, but arrived before the deadline of the last frame of the

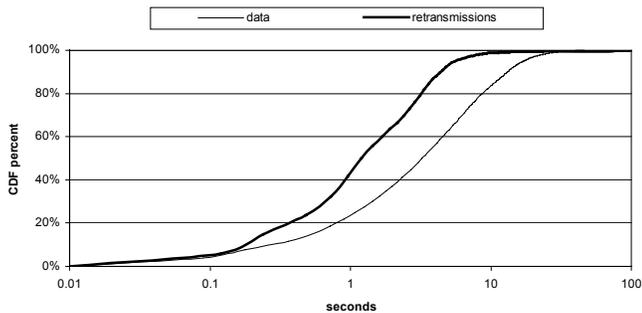


Figure 10. CDF functions of the amount of time by which retransmitted and data packets were late for decoding.

same GOP. Since the video decoder in our experiment could decompress frames at a substantially higher rate than the target fps, the client was able to use partially late packets for motion-compensated reconstruction of the remaining frames from the same GOP before *their* corresponding decoding deadlines. Out of 9,013 late retransmissions, 4,042 (49%) were partially late. Using each partially late packet, the client was able to save on average 4.98 frames from the same GOP¹² in D_{1p} and 4.89 frames in D_{2p} by employing the above-described catch-up decoding technique (for more discussion, see [22]).

The second type of delay, one-way delay jitter, caused 1,167,979 *data* (i.e., non-retransmitted) packets to miss their decoding deadlines. Hence, the total number of *underflow* (i.e., missing at the time of decoding) packets was $159,713 + 9,013 + 5,710 + 1,167,979 = 1,342,415$ (1.7% of the number of sent packets), which means that 98.9% of underflow packets were created by large one-way delay jitter, rather than by pure packet loss. Even if the clients had not attempted to recover any lost packets, still 73% of the missing packets at the time of decoding would have been caused by large delay jitter. Furthermore, these 1.3 million underflow packets caused a “freeze-frame” effect for the average duration of 10.5 seconds per ten-minute session in D_{1p} and 8.6 seconds in D_{2p} , which can be considered excellent given the relatively small startup delay used in the experiments.

To further understand the phenomenon of late packets, we plotted in Figure 10 the CDFs of the amount of time by which late packets missed their deadlines (i.e., the *additional* startup delay necessary to avoid a certain percentage of underflow events) for both late retransmissions and late data packets. As the figure shows, 25% of late retransmissions missed their deadlines by more than 2.6 seconds, 10% by more than 5 seconds, and 1% by more than 10 seconds (the tail of the CDF extends up to 98 seconds). At the same time, one-way delay jitter had a more adverse impact on data packets – 25% of late data packets missed their deadlines by more than 7 seconds, 10% by more than 13 seconds, and 1% by more than 27 seconds (the CDF tail extends up to 56 seconds).

The only way to reduce the number of late packets caused by both large RTTs and delay jitter is to apply a larger startup delay at the beginning of a session (in addition to freezing the display and adding extra startup delays during the session, which was not acceptable in our model). Hence, for example, Internet applications utilizing a 13-second startup delay (which corresponds to 10.3 seconds of *additional* delay in Figure 10)

¹² We used 10-frame GOPs in both sequences.

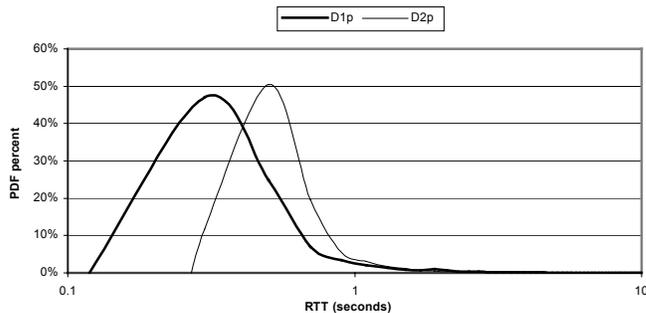


Figure 11. PDF functions of the RTT samples in each of D_{1p} and D_{2p} .

would be able to “rescue” 99% of late retransmissions and 84% of late data packets in similar streaming conditions.

VI. ROUND-TRIP DELAY

A. Overview

We should first mention that circuit-switched long-distance links through PSTN between our clients and remote access points did not significantly influence the measured end-to-end delays, because the additional delay on each long-distance link was essentially the propagation delay between New York and the location of the access point (which is determined by the speed of light and the geographical distance, i.e., 16 ms coast to coast). Clearly, this delay was negligible compared to the queuing and transmission delays experienced by each packet along the entire end-to-end path.

Figure 11 shows the PDF functions of the round-trip delay in each of D_{1p} and D_{2p} (660,439 RTT samples in both datasets). Although the tail of the combined distribution reached the enormous values of 126 seconds for simulated and 102 seconds for real retransmissions, the majority (75%) of the samples were below 600 ms, 90% below 1 second, and 99.5% below 10 seconds. The average RTT was 698 ms in D_{1p} and 839 ms in D_{2p} . The minimum RTT was 119 and 172 ms, respectively. Although very rare, extremely high RTTs were found in all six datasets $D_{1p}^a - D_{2p}^c$ (as a reminder, dataset D_{np}^x consists of successful sessions involving stream S_n and recorded through ISP_x). Furthermore, out of more than 660 thousand RTT samples in $\{D_{1p} \cup D_{2p}\}$, 437 were at least 30 seconds, 32 at least 50 seconds, and 20 at least 75 seconds.

Although pathologically high RTTs may seem puzzling at first, there is a simple explanation. Modem error correction protocols (i.e., the commonly used V.42) implement retransmission for corrupted blocks of data on the physical layer.¹³ Error correction is often necessary, if modems negotiate data compression (i.e., V.42bis) over the link, and is desirable, if PPP Compression Control Protocol (CCP) is enabled on the data-link layer. In all our experiments, both types of compression were enabled, imitating the typical setup of a home user. Therefore, if a client established a connection to a remote modem at a low bitrate (which was sometimes accompanied by a significant amount of noise in the phone line), each retransmission on the physical layer took a large time to complete before the data was delivered to the upper layers. In addition, large IP-

¹³ Since the telephone network beyond the local loop in the U.S. is mostly digital, we believe that dialing long-distance (rather than local) numbers had no significant effect on the number of bit errors during the experiment.

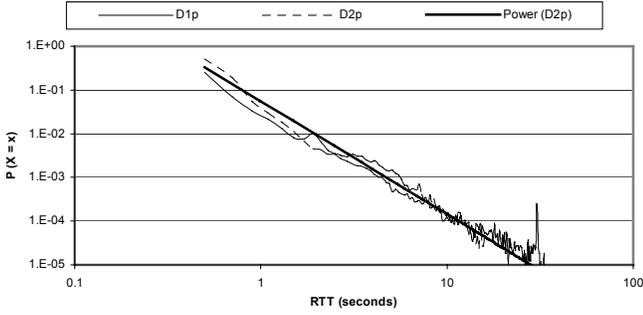


Figure 12. Log-log plot of the upper tails of the distribution of the RTT (PDF). The straight line is fitted to the PDF from D_{2p} .

level buffers on either side of the modem link further delayed packets arriving to or originating from the client host.

Note that the purpose of classifying sessions into failed and successful in section II.A was to avoid reporting pathological conditions caused by the modem links. Since only a handful (less than 0.5%) of RTTs in $\{D_{1p} \cup D_{2p}\}$ were seriously effected by modem-level retransmission and bit errors (we consider sessions with RTTs higher than 10 seconds to be caused by modem-related problems¹⁴), we conclude that our heuristic was successful in filtering out the majority of pathological connections and that future application-layer protocols, running over a modem link, should be prepared to experience RTTs in the order of several seconds.

Furthermore, the removal of sessions with RTTs higher than 10 seconds does not change any of the results below and, at the same time, prohibits us from showing the extent of variation in the network parameters experienced by a home Internet user. Therefore, since all sessions in $\{D_{1p} \cup D_{2p}\}$ were able to successfully complete, we consider the removal of sessions based on their large RTT to be unwarranted.

B. Heavy Tails

Mukherjee [17] reported that the distribution of the RTT along certain Internet paths could be modeled as a shifted gamma distribution. Even though the shape of the PDF in Figure 11 resembles that of a gamma function, the distribution tails in the figure decay much slower than those of an exponential distribution (see below).

Using our approach from section IV.D (i.e., assuming that each studied Internet random process is stationary), we extracted the upper tails of the PDF functions in Figure 11 and plotted the results on a log-log scale in Figure 12. The figure shows that a straight line (without loss of generality fitted to the PDF of D_{2p} in the figure) provides a good fit to the data (correlation 0.96) and allows us to model the upper tails of the PDF functions in Figure 12 as a Pareto distribution with PDF $f(x) = \alpha\beta^\alpha x^{-\alpha-1}$, where shape parameter α equals 1.16 in dataset D_{1p} and 1.58 in D_{2p} (as before, the distributions have a finite mean, but an infinite variance).

C. Variation of the RTT

We conclude the discussion of the RTT by showing that the round-trip delay exhibited a variation during the day similar to

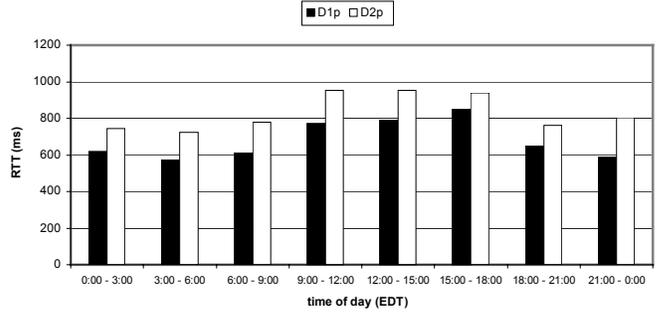


Figure 13. Average RTT as a function of the time of day.

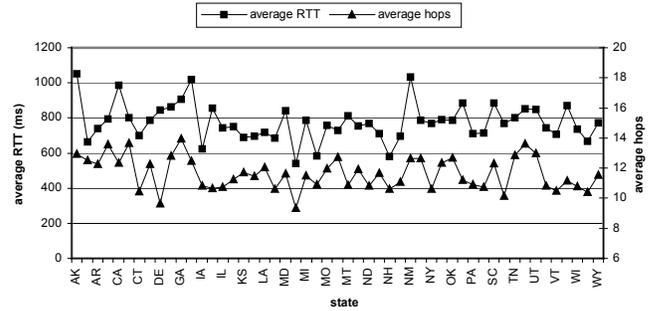


Figure 14. Average RTT and average hop count in each of the states in $\{D_{1p} \cup D_{2p}\}$.

that of packet loss (previously shown in Figure 5) and that the average RTT was positively correlated with the length of the end-to-end path. Figure 13 shows the average round-trip delay during each of the eight timeslots of the day (as before, each point in the figure represents the average of approximately 1,000 sessions). The figure confirms that the worst time for sending traffic over the Internet is between 9 am and 6 pm EDT and shows that the increase in the delay during the peak hours was relatively small (i.e., by only 30-40%).

Figure 14 shows the average RTT sampled by the clients in each of the 50 U.S. states. The average round-trip delay was consistently high (above 1 second) for three states – Alaska, New Mexico, and Hawaii. On the other hand, the RTT was consistently low (below 600 ms) also for three states – Maine, New Hampshire, and Minnesota. These results (except Minnesota) can be directly correlated with the distance from New York; however, in general, we find that the geographical distance of the access point from the East Coast had little correlation with the average RTT. Thus, for example, some states in the Midwest had small (600-800 ms) average round-trip delays and some states on the East Coast had large (800-1000 ms) average RTTs. A more substantial link can be established between the number of end-to-end hops and the average RTT as shown in Figure 14. Even though the average RTT of many states did not exhibit a clear dependency on the average length of the path, the correlation between the RTT and the number of hops in Figure 14 was reasonably high with $\rho = 0.52$. This result was intuitively expected since the RTT is essentially the sum of queuing and transmission delays at intermediate routers.

VII. PACKET REORDERING

A. Overview

Real-time protocols often rely on the assumption that packet

¹⁴ For example, one of the authors uses a modem access point at home with IP-level buffering on the ISP side equivalent to 6.7 seconds. Consequently, delays as high as 5-10 seconds may often be caused by non-pathological conditions.

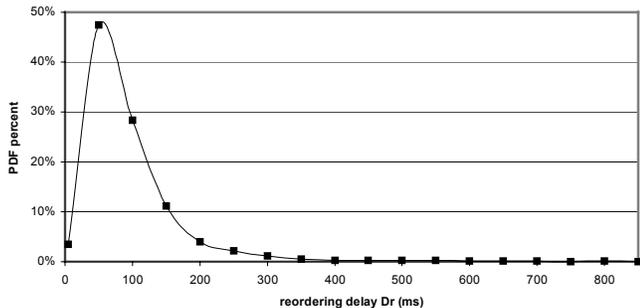


Figure 15. The PDF of reordering delay D_r in $\{D_{1p} \cup D_{2p}\}$.

reordering in the Internet is a rare and an insignificant event for all practical purposes (e.g., [8]). Although this assumption simplifies the design of a protocol, it also makes the protocol poorly suited for the use over the Internet. Certainly, there are Internet paths along which reordering is either non-existent or extremely low. At the same time, there are paths that are dominated by multipath routing effects and often experience reordering (e.g., Paxson [18] reported a session with 36% of packets arriving out of order).

Unfortunately, there is not much data documenting reordering rates experienced by IP traffic over modem links. Using intuition, we expected reordering in our experiments to be extremely rare given the low bitrates of streams S_1 and S_2 . However, we were surprised to find out that certain paths experienced consistent reordering with a relatively large number of packets arriving out of order, although the average reordering rates in our experiments were substantially lower than those reported by Paxson [18].

For example, in dataset D_{1p}^a , we observed that out of every three *missing*¹⁵ packets one was reordered. Hence, if users of ISP_a employed a streaming protocol, which used a gap-based detection of lost packets [8] (i.e., the first out-of-order packet triggers a NACK), 33% of NACKs would be flat-out redundant and a large number of retransmissions would be unnecessary, causing a noticeable fraction of ISP 's bandwidth to be wasted.

Since each missing packet is potentially reordered, the true frequency of reordering can be captured by computing the percentage of reordered packets relative to the total number of *missing* packets. The average reordering rate in our experiment was 6.5% of the number of *missing* packets, or 0.04% of the number of *sent* packets. These numbers show that our reordering rates were at least by a factor of 10 lower than those reported by Paxson [18], whose average reordering rates varied between 0.3% and 2% of number of sent packets depending on the dataset. This difference can be explained by the fact that our experiment was conducted at substantially lower end-to-end bitrates, as well as by the fact that Paxson's experiment involved several paths with extremely high reordering rates.

Out of 16,852 sessions in $\{D_{1p} \cup D_{2p}\}$, 1,599 (9.5%) experienced at least one reordering. Interestingly, the average session reordering rates in our datasets were very close to those in Paxson's 1995 data [18] (12% sessions with at least one reordering), despite the fundamental differences in sending rates. The highest reordering rate *per ISP* in our experiment occurred in D_{1p}^a , where 35% of the number of missing packets (0.2% of

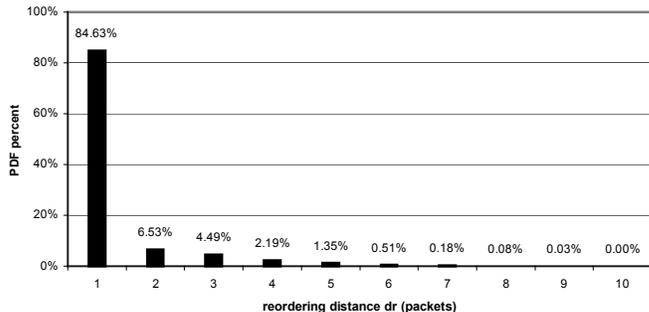


Figure 16. The PDF of reordering distance d_r in $\{D_{1p} \cup D_{2p}\}$.

the number of sent packets) turned out to be reordered. In the same D_{1p}^a , almost half of the sessions (47%) experienced at least one reordering event. Furthermore, the maximum number of reordered packets in a single session occurred in D_{1p}^b and was 315 packets (7.5% of the number of sent packets).

Interestingly, the reordering probability did not show any dependence on the time of day (i.e., the timeslot), and was virtually the same for all states.

B. Reordering Delay

To further study packet reordering, we define two metrics that allow us to measure the extent of packet reordering. First, let *packet reordering delay* D_r be the delay from the time when a reordered packet was declared as *missing* to the time when the reordered packet arrived to the client. Second, let *packet reordering distance* d_r be the number of packets (including the very first out-of-sequence packet, but not the reordered packet itself) received by the client during reordering delay D_r .

Figure 15 shows the PDF of reordering delay D_r in $\{D_{1p} \cup D_{2p}\}$. The largest reordering distance d_r in the combined dataset was 10 packets, and the largest reordering delay D_r was 20 seconds (however, in the latter case, d_r was only 1 packet). Although quite large, the maximum value of D_r is consistent with previously reported numbers (e.g., 12 seconds in Paxson's data [18]). The majority (90%) of samples in Figure 15 are below 150 ms, 97% below 300 ms, and 99% below 500 ms.

C. Reordering Distance

We next analyze the suitability of TCP's triple-ACK scheme in helping NACK-based protocols detect reordering. TCP's *fast retransmit* relies on *three* consecutive duplicate ACKs (hence the name "triple-ACK") from the receiver to detect packet loss and avoid unnecessary retransmissions. Therefore, if reordering distance d_r is either 1 or 2, the triple-ACK scheme successfully avoids duplicate packets, and if d_r is greater than or equal to 3, it generates a duplicate packet.

Figure 16 shows the PDF of reordering distance d_r in both datasets. Using the figure, we can infer that TCP's triple-ACK would be successful for 91.1% of the reordering events in our experiment, double-ACK for 84.6%, and quadruple-ACK for 95.7%. Note that Paxson's TCP-based data [18] show similar, but slightly better detection rates, specifically 95.5% for triple-ACK, 86.5% for double-ACK, and 98.2% for quadruple-ACK.

VIII. CONCLUSION

Our study showed that approximately half of all phone calls were unsuccessful and that typical U.S. users needed to re-dial connections on average once within the same state in order to

¹⁵ Missing packets are defined as gaps in sequence numbers.

sustain the minimum quality of service (QoS) that allowed them to receive low-bitrate Internet video. Our study found that an overwhelming majority of failures experienced by an end-user were caused by modem-to-modem pathologies, which included physical-layer connectivity problems (including failed negotiations and 14.4-19.2 kb/s negotiated bitrates), unresponsive modems that did not pick up the phone, busy phone lines, failed PPP negotiation (including failed user authentication, PPP timeouts, port disconnects by the remote modem, and various other PPP errors), failed traceroute to the server, and various IP-level problems that prevented UDP packets from bypassing the first-hop router inside the ISP network.

Interestingly, we found that large end-to-end delays did not pose much impediment to real-time retransmission during the experiment. The majority (94%) of “recoverable” lost packets returned to the client before their decoding deadlines. We also found that approximately 95% of all recovered packets were recovered using a single retransmission attempt (i.e., a single NACK). Nevertheless, we find that the current end-to-end delays in the dialup Internet are prohibitively high to support interactive real-time applications (such as video conferencing or telephony). For non-interactive applications, startup delays in the order of 10-15 seconds are recommended given a random access point used in streaming; however, startup delays as low as one second were found to be acceptable over certain paths during the night. Even using forward error correction (FEC) coding instead of retransmission to overcome packet loss does not allow us to substantially lower the startup delay, because one-way delay jitter in the dialup Internet is often very large.

We speculate that end-to-end delays under 100 ms will be required to support interactive streaming, which seems to be currently possible with DSL and certain cable modems. Furthermore, we believe that with broadband access at home, the performance of real-time streaming will largely depend on the end-to-end congestion control employed in the streaming protocol, rather than on the backbone Internet packet-loss rates, a particular retransmission scheme, or delay jitter (all of which are significantly less relevant given low end-to-end delays). Hence, in the future, it is extremely important to develop congestion control suitable for real-time multimedia flows that scales to a large number of concurrent users and can be employed incrementally with the existing TCP flows (i.e., it should possess some form of TCP-friendliness).

ACK-based congestion control [12] for TCP-like flows is understood fairly well, but it is still not clear whether ACK-based flow control is suitable for real-time flows. On the other hand, even though NACK-based flow control is great for rate-based applications, its “open-loop” operation makes it much more unstable and less scalable [13]. We believe that future research should first address these congestion control issues before real-time streaming becomes widely available in the Internet.

REFERENCES

- [1] A. Acharya and J. Saltz, “A Study of Internet Round-trip Delay,” *Technical Report CS-TR-3736*, University of Maryland, December 1996.
- [2] H. Balakrishnan, V.N. Padmanablah, S. Seshan, M. Stemm, and R.H. Katz, “TCP Behavior of a Busy Internet Server: Analysis and Improvements,” *IEEE INFOCOM*, March 1998.
- [3] J. Bolliger, T. Gross, and U. Hengartner, “Bandwidth Modelling for Network-Aware Applications,” *IEEE INFOCOM*, March 1999.
- [4] J-C. Bolot and T. Tuletto, “Experience with Rate Control Mechanisms for Packet Video in the Internet,” *ACM Computer Communication Review*, January 1998.
- [5] J-C. Bolot, “End-to-End Packet Delay and Loss Behavior in the Internet,” *ACM SIGCOMM*, September 1993.
- [6] M.S. Borella, D. Swider, S. Uludag, and G.B. Brewster, “Internet Packet Loss: Measurement and Implications for End-to-End QoS,” *International Conference on Parallel Processing*, August 1998.
- [7] R. Caceres, P.B. Danzig, S. Jamin, D. Mitzel, “Characteristics of Wide-Area TCP/IP Conversations,” *ACM SIGCOMM*, 1991.
- [8] B. Dempsey, J. Liebeherr, and A. Weaver, “On Retransmission-based Error Control for Continuous Media Traffic in Packet-switching Networks,” *Computer Networks and ISDN Systems*, vol. 28, no. 5, March 1996.
- [9] S.P. Pizzo, “Why Is Broadband So Narrow?” *Forbes ASAP*, September 2001, p. 50.
- [10] ISP Planet and Cahners In-Stat Group, “Dial-Up Remains ISPs' Bread and Butter,” http://isp-planet.com/research/2001/dialup_butter.html, 2001.
- [11] ISP Planet and Telecommunications Research International, “U.S. Residential Internet Market Grows in Second Quarter,” http://isp-planet.com/research/2001/us_q2.html, 2001.
- [12] V. Jacobson, “Congestion Avoidance and Control,” *ACM SIGCOMM*, 1988.
- [13] D. Loguinov and H. Radha, “Increase-Decrease Congestion Control for Real-time Streaming: Scalability,” *IEEE INFOCOM*, June 2002.
- [14] D. Loguinov and H. Radha, “On Retransmission Schemes for Real-time Streaming in the Internet,” *IEEE INFOCOM*, April 2001.
- [15] A. Mena and J. Heidemann, “An Empirical Study of Real Audio Traffic,” *IEEE INFOCOM*, March 2000.
- [16] J.C. Mogul, “Observing TCP Dynamics in Real Networks,” *ACM SIGCOMM*, 1992.
- [17] A. Mukherjee, “On the Dynamics and Significance of Low-Frequency Components of Internet Load,” *Internetworking: Research and Experience*, vol. 5, no. 4, December 1994.
- [18] V. Paxson, “Measurements and Analysis of End-to-End Internet Dynamics,” *Ph.D. dissertation*, Computer Science Department, University of California at Berkeley, 1997.
- [19] H. Radha, Y. Chen, K. Parthasarathy, and R. Cohen, “Scalable internet video using MPEG-4,” *Signal Processing: Image Communication*, 1999.
- [20] RealPlayer, Real Networks, <http://www.real.com>.
- [21] A.R. Reibman and B.G. Haskell, “Constraints on Variable Bitrate Video for ATM Networks,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 4, December 1992.
- [22] I. Rhee, “Error Control Techniques for Interactive Low Bitrate Video Transmission over the Internet,” *ACM SIGCOMM*, September 1998.
- [23] W. Tan and A. Zakhor, “Real-Time Internet Video Using Error Resilient Scalable Compression and TCP-Friendly Transport Protocol,” *IEEE Transactions on Multimedia*, vol. 1, no. 2, June 1999.
- [24] UUnet Latency Statistics. <http://uunet.com/network/latency>.
- [25] M. Yajnik, S. Moon, J. Kurose, and D. Townsley, “Measurement and Modelling of the Temporal Dependence in Packet Loss,” *IEEE INFOCOM*, March 1999.
- [26] Y. Zhang, V. Paxson, and S. Shenker, “The Stationarity of Internet Path Properties: Routing, Loss, and Throughput,” *ACIRI Technical Report*, May 2000, <http://www.aciri.org/vern/papers.html>.