Multipath Live Streaming via TCP: Scheme, Performance and Benefits

BING WANG, WEI WEI, and ZHENG GUO
University of Connecticut, USA
DON TOWSLEY
University of Massachusetts, Amherst, USA

Motivated by the wide use of TCP for multimedia streaming in practice and the increasing availability of multipath between end hosts, we study multipath live streaming via TCP in this article. We first design a simple and practical TCP-based multipath streaming scheme, named *Dynamic MPath-streaming (DMP-streaming)*, which dynamically distributes packets over multiple paths by *implicitly inferring* the available bandwidths on these paths. To allow systematic performance study, we develop an analytical model for DMP-streaming and validate the model using extensive *ns* simulation and Internet experiments. We explore the parameter space of this model and find that DMP-streaming generally provides satisfactory performance when the aggregate achievable TCP throughput is 1.6 times the video bitrate, when allowing a few seconds of startup delay. Last, we comment on the benefits of using multipath versus single path for TCP-based streaming.

Categories and Subject Descriptors: C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Application (multimedia streaming)*; H.1.0 [**Models and Principles**]: General

General Terms: Performance

Additional Key Words and Phrases: Performance modeling, multimedia streaming

ACM Reference Format:

Wang, B., Wei, W., Guo, Z., and Towsley, D. 2009. Multipath live streaming via TCP: Scheme, performance and benefits. ACM Trans. Multimedia Comput. Commun. Appl. 5, 3, Article 25 (August 2009), 23 pages.
DOI = 10.1145/1556134.1556142 http://doi.acm.org/10.1145/1556134.1556142

1. INTRODUCTION

The increasing availability of multipaths between end hosts has motivated a number of recent studies on multipath audio/video streaming (i.e., streaming over multiple network paths) [Golubchik et al. 2002; Apostolopoulos et al. 2002; Liang et al. 2001; Nguyen and Avideh 2004; Ribeiro et al. 2005].

The first version of this article appeared in *Proceedings of CoNEXT* 2007 [Wang et al. 2007].
This work was supported in part by UCONN large grant, the National Science Foundation under ANI-0325868, ANI-0240487, ANI-0085848, and EIA-0080119, and NSF CAREER award 0746841. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.
Authors' addresses: B. Wang, W. Wei, and Z. Guo, Computer Science and Engineering Department, University of Connecticut, Storrs, CT 06269; email: {bing, weiwei, guozheng}@engr.uconn.edu; D. Towsley, Department of Computer Science, University of Massachusetts, Amherst, MA 01003; email: towsley@cs.umass.edu.

All these studies assume UDP as the transport protocol. Indeed, TCP is conventionally regarded as inappropriate for multimedia streaming, since its backoff and retransmission mechanisms may lead to long delays which violate the real-time requirement of multimedia streaming.

In this article, defying the conventional wisdom, we study an approach that relies on TCP for multipath streaming. This is motivated by the wide use of TCP for streaming in practice and our earlier results on single-path TCP streaming [Wang et al. 2004]. TCP streaming is widely supported in commercial streaming products (e.g., Real Media and Windows Media). Furthermore, recent measurement studies have shown that, for both stored-video and live streaming, a significant fraction of the traffic (around or above 50%) uses HTTP/TCP [Wang et al. 2001; Sripanidkulchai et al. 2004; van der Merwe et al. 2002]. In our earlier work [Wang et al. 2004], we studied the performance of single-path TCP streaming and found that its performance is generally satisfactory when the achievable TCP throughput is roughly twice the media bitrate, when allowing a few seconds of startup delay. This result partly explains why TCP streaming has been an attractive option in practice: When the last-mile access link is the bottleneck, such a bandwidth requirement can be satisfied even by broadband connections (cable or ADSL) for a large fraction of streaming multimedia clips in the Internet today.

Motivated by the previous observations, we focus on multipath live streaming via TCP in this article. More specifically, we consider the scenario in which a video server generates content in real time and streams it via TCP to a client over K paths. These K paths may or may not share bottleneck links. We seek to answer the following questions: *Under what circumstances can multipath TCP-based live streaming provide satisfactory performance? What are the benefits from using multiple paths versus a single path in TCP-based live streaming?*

To answer the preceding questions, we first design a simple and practical multipath TCP-based streaming scheme, named *Dynamic MPath-streaming (DMP-streaming)*. It dynamically distributes packets over the multiple paths (to accommodate bandwidth fluctuations) by *implicitly inferring* the available bandwidths on these paths. It is simple in that it requires no explicit probing/feedback on network bandwidths, and does not exploit any error recovery (e.g., using redundant packets and/or coding) or rate adaptation at the application level. On the other hand, it generally outperforms round-robin packet allocation [Tullimas et al. 2008] (see Section 3). We use DMP-streaming as a baseline scheme and study its performance through an analytical model. Our main contributions are as follows.

—We develop an analytic model for DMP-streaming and validate the model using extensive *ns* simulation and Internet experiments. The analytical model allows a *systematic* study, a task that is difficult when using empirical measurements or simulation alone. To the best of our knowledge, this is the first analytical model on multipath streaming via TCP.

—We systematically vary the parameters in this model to explore the parameter space when using two paths in DMP-streaming. We find that the performance of DMP-streaming is not sensitive to path heterogeneity. Furthermore, performance is generally satisfactory when the *aggregate* achievable TCP throughput is 1.6 times the video bitrate, when allowing a few seconds of startup delay.

Our results help answer the following two important questions on TCP-based streaming: (i) If a video bitrate is satisfied by a single path, can two paths, each with half of the achievable TCP throughput of the single path, support the same video bitrate? When the access link is the bottleneck, this question transforms to: Can a high-bandwidth access link be replaced by two lower-bandwidth links while maintaining the same streaming performance? (ii) If a video bitrate is satisfied by a single path, can two such paths support videos with twice the bitrate? When the access link is the bottleneck, this question transforms to: If a user subscribes to two access networks of similar bandwidths (e.g., ADSLs from two different providers), can he/she view videos with bitrates twice as those supported by a single access network?

Our results indicate that the answer to both of the previous questions is: yes. This is because multipath TCP streaming provides satisfactory performance when the ratio of the aggregate achievable TCP throughput over the video bitrate exceeds 1.6, lower than the ratio of 2 in single-path TCP streaming [Wang et al. 2004]. Therefore, for question (i), two paths, each with half of the achievable TCP throughput of the single path, can support the same (even higher) video bitrate supported by the single path; for question (ii), two paths, each with the achievable TCP throughput of the single path, can support videos with twice (even more than twice) the bitrate supported by the single path.

The rest of the article is organized as follows. Section 1.1 summarizes related work. Section 2 describes the problem setting. Sections 3 and 4 present DMP-streaming and its analytical model, respectively. Sections 5 and 6 describe validation of the model using *ns* simulation and Internet experiments, respectively. Section 7 explores the parameter space of the model. Section 8 compares DMP-streaming and a static streaming scheme. Finally, Section 9 concludes the article and presents future work.

1.1 Related Work

Multipath continuous media streaming is studied in Golubchik et al. [2002], Apostolopoulos et al. [2002], Liang et al. [2001], Nguyen and Avideh [2004], Ribeiro et al. [2005], Jurca and Frossard [2006], and Chesterfield et al. [2005]. In particular, Golubchik et al. [2002] and Ribeiro et al. [2005] demonstrate the benefits of using multiple paths for continuous media streaming. In Apostolopoulos et al. [2002] and Liang et al. [2001], coding techniques (multiple description codes) are applied to the video streams to improve loss recovery. The study in Nguyen and Avideh [2004] determines the sending rate and the distribution of packets on the multiple paths to minimize the loss rate at the receiver. The work in Jurca and Frossard [2006] proposes a heuristic algorithm for multipath video streaming that provides close-to-optimal performance. The studies in Chesterfield et al. [2005] and Sharma et al. [2008] propose multipath streaming schemes suitable for cellular links and wireless mesh networks, respectively. All the aforesaid studies use UDP as the transport protocol. We study multipath streaming via TCP, which is fundamentally different from UDP-based streaming (e.g., UDP-based streaming may not provide error recovery and/or congestion control; even if it provides such functionalities, the mechanisms are very different from those in TCP). Our scheme, DMP-streaming, is mainly designed for wired networks. It is very simple; more sophisticated schemes (e.g., providing rate adaptation and/or error recovery at the application level) may lead to better performance, which are, however, beyond the scope of our study.

Using TCP for multimedia streaming eliminates the need for error recovery at the application level and automatically provides TCP-friendliness. Furthermore, it is more likely to pass through firewalls in practice. These advantages have motivated a number of studies on TCP-based streaming. The studies of Seelam et al. [2001], Krasic and Walpole [2001], de Cuetos and Ross [2002], and de Cuetos et al. [2002] focus on how to adapt the video bitrate to deal with network bandwidth fluctuations. Our earlier work [Wang et al. 2004] studied the performance of multimedia streaming using TCP. The study in Kim and Ammar [2006] analytically determines the proper receiver buffer size to ensure a prescribed video quality for TCP streaming. All the previous studies focus on one TCP flow on a single path, while we focus on multiple TCP flows in this article. A recent study [Tullimas et al. 2008] proposes a receiver-driven multimedia streaming scheme using multiple TCP connections. The scheme provides resilience against short-term insufficient bandwidth by controlling the receiver window sizes and the number of TCP connections. Our scheme does not adjust receiver window sizes or the number of TCP connections. Instead, our main contribution is an analytic study of DMP-streaming to identify the conditions under which it provides satisfactory performance.

The literature on TCP modeling is extensive. Much of the TCP modeling focuses on TCP performance for file transfers, assuming long-lived flows [Mathis et al. 1997; Padhye et al. 1998, 1999; Altman et al.

2000; Figueiredo et al. 2002] or short-lived flows [Cardwell et al. 2000; Mellia et al. 2002]. The study of Bohacek [2003] models the TCP congestion window to determine a TCP-friendly transmission rate for UDP video flows. Our study differs from the preceding in that we develop analytical models for multipath TCP streaming, with a focus on multipath and the timeliness of the packets.

Last, Hsieh and Sivakumar [2002] point out several limitations of using TCP over multiple paths for reliable data transfer when the access network has high bandwidth fluctuations. Our study is in the context of a wired network (that has relatively stable bandwidth) for multimedia streaming (that can tolerate a certain amount of packet loss).

2. PROBLEM SETTING

Suppose that a client is connected to a server using K paths ($K \geq 2$), indexed 1 to K. These paths are formed using a multipath architecture (e.g., multihoming of the end hosts). They may or may not share bottleneck links. We consider live streaming, that is, the server generates video content in real time and is only able to transmit content that has already been generated. The server opens K TCP connections, one on each path, and stripes the generated video packets using the multiple TCP flows to the client. We assume that the server does not send redundant packets or drop packets. Each packet carries an ID that corresponds to its position in the video. Packets on the same path arrive in increasing order of their IDs (since TCP provides in-order transmission). However, packets with higher IDs on one path may arrive earlier than packets with lower IDs on another path. These out-of-order arrivals do not cause any problem during playback since all packets arriving earlier than their playback times are stored in a local buffer at the client; at the time to playback a packet, the client fetches the packet from the local buffer. We assume this buffer to be sufficiently large so that no packet is lost at the client side. This assumption is reasonable since modern machines are equipped with a large amount of storage. Last, the client allows a startup delay, τ, that is on the order of seconds. Note that we do not consider interactive streaming applications (e.g., video conference), which requires a much shorter startup delay.

We consider a Constant BitRate (CBR) video, motivated from measurement studies that conclude that most videos streamed over the Internet are CBR [Li et al. 2003]. Let μ denote the playback rate of the video (in packets per second). For simplicity, all packets are assumed to be of the same size. For analytical tractability, we assume continuous playback at the client. A packet arriving later than its playback time is referred to as a *late packet*. Under our assumption of continuous playback at the client, a late packet will be skipped by the client and may lead to a glitch during playback.[1] Therefore, we use the *fraction of late packets*, that is, the probability that a packet is late, as our performance metric. Strictly speaking, the fraction of late packets does not correspond directly to viewing quality, since human perception tolerates occasional glitches in the playback. However, we are not aware of any quantitative metric that directly corresponds to the viewing quality perceived by a human. We therefore use the fraction of late packets to roughly measure streaming performance.

We next describe two properties of live streaming. The first property is that the number of *early packets*, namely, packets arriving earlier than their playback times, is bounded from above. The second property is related to TCP throughputs on the multiple paths.

2.1 Number of Early Packets in Live Streaming

Figure 1 illustrates multipath live streaming via TCP. The video server generates packets at a constant rate equal to the playback rate of the video, that is, μ packets per second. For ease of exposition, we

[1]An alternative way to deal with late packets is that the client will pause and buffer for a certain amount of time, and then resume playback. We consider continuous playback instead of this pause-and-buffer type of playback for analytical tractability.

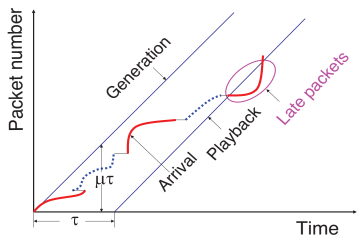


Fig. 1. Illustration of multipath live streaming via TCP. In this example, $K = 2$, solid and dashed curves represent packet arrivals from the first and the second path respectively.

assume that packets are generated from time 0, starting with packet number 0. Let $G(t)$ denote the number of packets generated at the server by time t. Then $G(t) = \mu t$. The client starts to playback the video at time τ. Let $B(t)$ denote the number of packets played back by the client by time t. Then $B(t) = \mu(t - \tau)$ since the video is CBR-coded, $t \geq \tau$. Observe that $G(t) - B(t) = \mu\tau$. Since only packets that are generated can be transmitted, the total number of packets arriving at the client by time t is at most $G(t)$. Therefore, the number of early packets is at most $G(t) - B(t) = \mu\tau$ at any time t. This property is used in our model for multipath live streaming via TCP in Section 4.

2.2 TCP Throughputs in Live Streaming

Let σ_k denote the average *achievable TCP throughput* (in packets per second) on path k, which is the throughput achieved by a greedy TCP source, that is, a source that always has data to transmit. Let σ'_k denote the average TCP throughput (in packets per second) on path k in live streaming. Then $\sigma'_k \leq \sigma_k$ because the TCP source on the k-path may not always have data to send (data are generated in real time and only generated packets can be transmitted). Furthermore, $\sum_{i=1}^{K} \sigma'_k \leq \mu$ since the packet generation rate is μ.

3. A SCHEME FOR MULTIPATH STREAMING VIA TCP

In live-streaming via TCP, to reduce the number of late packets, the server needs to transmit the generated packets to the client as fast as allowed by the TCP flows on the multiple paths. This is because network congestion may cause the aggregate TCP throughput to be occasionally lower than the video playback rate; buffering as many packets as possible can compensate for this adverse effect. We next describe our scheme, *Dynamic MPath-streaming (DMP-streaming)*, and then compare it with a round-robin multipath streaming scheme.

3.1 DMP-Streaming

A key question in multipath streaming is: Given the multiple paths, which path should a packet be assigned to? Intuitively, a desired streaming scheme allocates packets over the multiple paths dynamically according to the current throughputs of these paths and allocates more packets onto paths with higher throughputs (as in existing UDP-based schemes, e.g., Rejaie and Ortega [2003]). Furthermore, it should avoid explicitly probing for bandwidth on each path (so that no probing traffic is generated). We next describe DMP-streaming and show that it satisfies the aforestated desired properties.

Figure 2 describes the actions of the video server and the TCP senders in DMP-streaming. The server places the generated video packets into a queue, referred to as the *server queue*, in the order of their playback times. The TCP senders on each of the paths can fetch packets from the server queue. However, at a certain point of time, only one TCP sender is allowed to access the server queue (this can be achieved through a locking mechanism). More specifically, when a TCP sender can send data, it first

```
At the video server:
  Generate packets
  Place the generated packets into server queue

At a TCP sender:
  If (it can send packets) {
    if (it obtains access to the server queue) {
      do {
        Fetch packets from the head of the server queue
      } till (it cannot send or the server queue is empty)
    }
  }
```

Fig. 2. DMP-streaming: actions of the video server and the TCP senders.

obtains access to the server queue, and then fetches packets from the head of the server queue until it cannot send any more packets (i.e., this TCP sender is blocked) or when no packet is inside the server queue. At that time, it releases its lock on the server queue so that another TCP source can access the server queue.

We now demonstrate that DMP-streaming has the desired properties that are described earlier. First, it clearly allocates packets in a dynamic manner over the multiple paths (each TCP source fetches packets dynamically from the server queue). Second, it allocates more packets to the paths with higher achievable TCP throughputs by *implicitly* inferring the achievable TCP throughputs on these paths. This can be explained as follows. In the current implementation of TCP, a TCP sender places packets in its sending buffer before transmitting them into the network. The TCP sender cannot send any more packets (i.e., it blocks) when the sending buffer is full. Therefore, once the TCP sending buffers on the multiple paths becomes full after the initial transient period, a path with a higher achievable TCP throughput drains packets from its sending buffer faster and fetches more packets from the server queue. Therefore, a TCP source on a path with a higher achievable throughput sends a larger fraction of the packets.

As we can see, DMP-streaming is extremely simple; it takes advantage of the congestion control mechanisms in TCP to adapt to bandwidth fluctuations in the network paths. It can be used regardless of whether the multiple TCP flows share bottleneck links or not. Furthermore, it is also applicable to stored-video streaming. We focus on using DMP-streaming for live streaming in this article; its performance for stored-video streaming is left as future work.

3.2 Comparison with an Alternative Scheme

We next compare DMP-streaming and a round-robin multipath streaming scheme. The round-robin scheme is similar to the scheme in Tullimas et al. [2008] and works as follows. When the server has packets to send, it cycles through the TCP connections in a round-robin fashion. If a TCP connection cannot send (i.e., it is blocked), then the server chooses the next TCP connection. We refer to this round-robin scheme as *RR-streaming*. RR-streaming is similar to DMP-streaming in that it implicitly infers the available bandwidth. However, it differs from DMP-streaming in that it allocates packets to the multiple paths as evenly as possible, while DMP-streaming allocates more packets to paths with higher bandwidths.

We compare the performance of RR-streaming and DMP-streaming through *ns* simulation. The topology is shown in Figure 3. A video server streams a video to a client via TCP over two paths. The video length is 10,000 seconds. Its bitrate is 600Kbps (playback rate is 50 packets per second and each packet

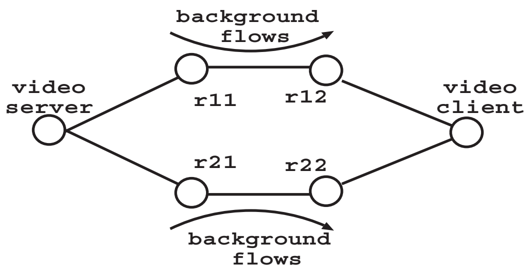
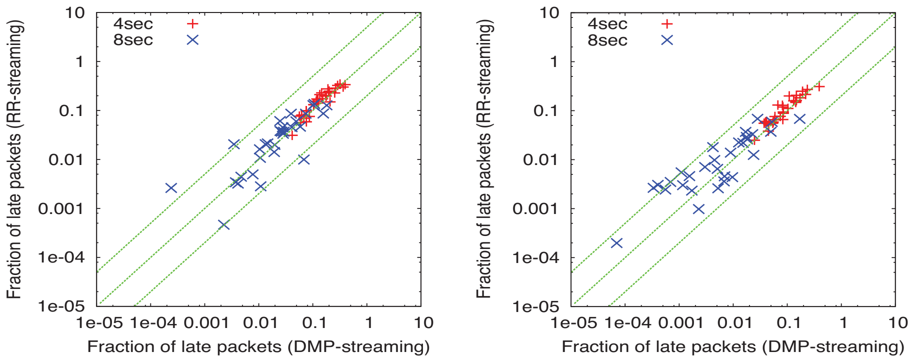


Fig. 3. Simulation setting in *ns*: the video server spreads the video over two paths to the client. Packet losses are caused by buffer overflows on the bottleneck links from router r_{k1} to r_{k2}, $k = 1, 2$.



(a) Setting 1. (b) Setting 2.

Fig. 4. DMP-streaming versus RR-streaming. The two paths are homogenous in Setting 1 and heterogeneous in Setting 2.

is 1500 bytes). On path k, link (r_{k1}, r_{k2}) is the bottleneck link, where packets are lost due to buffer overflow, $k = 1, 2$. The link from the video server to r_{k1} (and the link from r_{k2} to the video client) has propagation delay of 10 ms and bandwidth of 100Mbps. We consider two settings. In Setting 1, the two paths are homogeneous. In other words, the bottleneck links on these two paths are the same: on each path, 9 FTP flows and 40 HTTP flows traverse the bottleneck link, and the bottleneck link has a propagation delay of 40 ms, bandwidth of 3.7Mbps, and a buffer to hold 50 packets. Setting 2 differs from Setting 1 in that the two paths are heterogeneous: the propagation delay of the bottleneck link on the second path is reduced to 1 ms. We make 30 runs using independent random seeds for each setting. In Setting 1, the average achievable TCP throughput on each path is 376Kbps (measured from long-live FTP flows). In Setting 2, the average achievable TCP throughputs on the two paths are 376Kbps and 427Kbps, respectively. Figures 4(a) and 4(b) plot the results under DMP-streaming versus those under RR-streaming in log-log scale for Settings 1 and 2, respectively. In both figures, the 45-degree line starting at the origin represents that the fractions of late loss under the two schemes are the same; along the upper and lower 45-degree lines, the fraction of late packets under RR-streaming is respectively 5 times higher and lower than that in DMP-streaming. Each figure plots the results for 30 runs, with the startup delays of 4 and 8 seconds. We observe that for homogeneous paths, DMP-streaming outperforms RR-streaming in, respectively, 22 and 19 runs for startup delays of 4 and 8 seconds. For heterogenous paths, DMP-streaming outperforms RR-streaming in, respectively, 26 and 22 runs for startup delays of 4 and 8 seconds. Therefore, DMP-streaming generally outperforms RR-streaming under both homogeneous and heterogeneous paths. Furthermore, as expected, the advantage of DMP-streaming is more prominent when the paths are heterogenous.

Note that the previous comparison uses the fraction of late packets as the performance metric. RR-streaming tends to distribute consecutive packets on different paths, and hence may lead to less bursty late losses. We leave further comparison of DMP-streaming and RR-streaming using other performance metrics as future work. In the rest of the article, we use DMP-streaming as a baseline scheme to identify the conditions under which multipath streaming leads to satisfactory performance.

4. ANALYTICAL MODEL

We develop a continuous-time Markov model for DMP-streaming. As we shall see, this model allows us to *systematically* vary the various parameters to explore the performance of DMP-streaming (Section 7). In the following, we first provide an overview of this model and then describe it in detail.

4.1 Model Overview

Our model assumes a single TCP flow on each path from server to client. It is developed by considering the data production and consumption at the client-side buffer: the multiple TCP connections from the server to the client produce (transmit) packets and store them in the client-side buffer; the client starts to consume (i.e., play back) packets in the buffer from time τ at a constant rate of μ packets per second. We add the constraint that a producer stops producing packets when there are $N_{max} := \mu\tau$ early packets in the buffer. This follows from an earlier observation that the number of early packets in the client-side buffer never exceeds N_{max} (see Section 2.1).

Note that, although out-of-order packet arrivals from the multiple paths does not lead to problems during playback (they are placed in their correct positions in the client's local buffer), they lead to challenges in modeling DMP-streaming. One way to deal with out-of-order packets is to include the packet number of each packet in the model. However, this will make the state space of the model prohibitively large and render the model intractable. On the other hand, in DMP-streaming, as confirmed by our simulation and experimental results (in Sections 5 and 6), playing back packets in their arriving order leads to a similar fraction of late packets as when playing them back according to their playback times. The reasons can be explained by considering the following two cases. In both cases, we consider an arbitrary packet, i, arriving on path k.

—*Case* 1. Packet i is not late (i.e., it arrives earlier than its playback time). Suppose packet j, $j > i$, arrives earlier than packet i and is played back as packet i when playing back packets in their arriving order. This case, however, does not cause an error to the fraction of late packets (since neither packet i nor j is late).

—*Case* 2. Packet i is late (i.e., path k is congested). In particular, suppose a sequence of n packets on path k are late ($n \geq 1$). If there are out-of-order packets and packets are played back in their arriving order, packets from another path may be played back as these n packets, which causes an error. However, when this happens, we expect n to be very small and hence the error is negligible. This is because DMP-streaming reduces the number of packets sent on a path when the path becomes congested. Since the startup delays are much longer than the round-trip times of the TCP flows (a few seconds versus a few hundred milliseconds), we expect the number of packets sent on a congested path to have been reduced significantly (i.e., n is small) when late packets occur.

In our model, we make a simplifying assumption that packets are played back in their arrival order (note that this is only for modeling convenience; the client plays back packets according to their playback times). Under this simplifying assumption, we only need to keep track of the number of early packets in the model. We next describe the model in detail.

4.2 Model for DMP-Streaming

Let $(X_1(t), \ldots, X_K(t), N(t))$ represent the state of the model for DMP-streaming at time t, where $X_k(t)$ is the state of the kth TCP flow and $N(t)$ is the number of early packets in the client's local buffer at time t, $k = 1, \ldots, K$, $t \geq 0$. The state transition of the model is governed by the state transitions of the various TCP flows and the packet consumption event. In the following, we first describe the state transition for each TCP flow. We then describe the evolution of $N(t)$ and how to obtain the fraction of late packets from the model.

The state transitions of the different TCP flows are independent of each other. The state of the kth TCP flow at time t, $X_k(t)$, is a tuple containing five components, namely, $X_k(t) = (W_k(t), C_k(t), L_k(t), E_k(t), Q_k(t))$, where $W_k(t)$ is the window size; $C_k(t)$ models the delayed ACK behavior of TCP (it changes from 0 to 1 or from 1 to 0 after a state transition); $L_k(t)$ is the number of packets lost when the previous transition occurs; $E_k(t)$ denotes whether the connection is in a timeout state and the value of the back-off exponent; $Q_k(t)$ indicates whether a packet being sent in the timeout phase is a retransmission ($Q_k(t) = 1$) or a new packet ($Q_k(t) = 0$). The transition rate from one state to another depends on the states and the parameters of this TCP flow, including its RTT, loss rate, and timeout value.[2] A detailed description of the state transition rate for each TCP flow is found in Appendix A.

We now describe how the number of early packets, $N(t)$, evolves over time. The state of the Markov chain changes under two types of events: (1) when any of the TCP flows makes a transition, and (2) when a packet is consumed (played back) from the client's local buffer. The first type of events increases $N(t)$ while the second type of events decreases $N(t)$. To satisfy the constraint that $N(t) \leq N_{max} = \mu\tau$ in live streaming (see Section 2.1), a TCP flow does not make a transition if the current number of early packets is N_{max}. Let $\mathcal{E}(t)$ denote the type of event that triggers the transition at time t, $\mathcal{E}(t) = \mathcal{P}$ if it is a TCP event, and $\mathcal{E}(t) = \mathcal{C}$ if it is a packet consumption event. Considering these two conditions, we have the following.

—*Condition* 1 ($\mathcal{E}(t) = \mathcal{P}$). Suppose the kth TCP flow triggers the transition. Then the number of early packets, $N(t)$, is increased by $s_k(t)$ (not exceeding $N_{max} = \mu\tau$), where $s_k(t)$ is the number of packets that the kth TCP flow transmits successfully at the transition (details in Appendix A).

—*Condition* 2 ($\mathcal{E}(t) = \mathcal{C}$). Then the number of early packets, $N(t)$, is reduced by 1.

Note from the preceding that, in our model, a flow with a higher achievable throughput contributes more early packets, and hence captures the property of DMP-streaming that such a flow transmits a larger fraction of packets.

For sufficiently long videos, we obtain the fraction of late packets from the stationary distribution of $N(t)$ as

$$f = \lim_{t \to \infty} P(N(t) < 0 \mid \mathcal{E}(t) = \mathcal{C}). \tag{1}$$

We numerically solve for the stationary distribution of $N(t)$ using the TANGRAM-II modeling tool [de Souza e Silva and Leao 2000].

The previous model assumes that loss events over the multiple paths are independent. This is true when the TCP flows do not share bottleneck links. When the TCP flows share bottleneck links, as long as the losses in the TCP flows are not significantly correlated, our model may still provide accurate results. We validate our model for both cases in Section 5.

[2]Our assumption of loss process follows Padhye et al. [1999] and Figueiredo et al. [2002]; that is, we assume packet losses are caused by congestion. In particular, packet losses in different RTTs are independent and packet losses in the same RTT are correlated (if a packet is lost, all remaining packets until the end of the RTT are lost). Last, the effect of lost ACKs is regarded as negligible.

Table I. Configurations of the Bottleneck Link

Config.	FTP flows	HTTP flows	Prop. Delay (ms)	Bandwidth (Mbps)	Buffer (pkts)
1	9	40	40	3.7	50
2	9	40	1	3.7	50
3	19	40	40	5.0	50
4	5	20	1	5.0	30

5. MODEL VALIDATION USING *NS*

In this section, we validate our model for DMP-streaming using *ns*. We use $K = 2$, that is, two TCP flows are used for live streaming. In the following, we first describe our methodology for validation and then describe the validation results.

5.1 Methodology

We refer to the TCP flows that are used to stream video as *video streams*. The network is simulated as follows. Each video stream traverses a path with a bottleneck link. This bottleneck link is also used by multiple FTP and HTTP flows, referred to as *background flows*. We simulate four different configurations for the bottleneck link on a path by varying the delay, bandwidth, and buffer size of that link and the number of background flows traversing that link. These configurations are listed in Table I. They are not intended to mimic the actual settings in practice. Rather they are chosen to provide a wide range of round-trip times, different loss rates, and timeout values to the video streams.

For the video stream (via TCP) on the kth path, let p_k, R_k and R^k_{TO} denote respectively the loss probability, the round-trip time (RTT), and the first retransmission timer. We further define $T_{O_k} = R^k_{TO}/R_k$, which reflects the variation of the RTTs. In all cases, the video streams uses TCP Reno. We are interested in the steady-state behavior of multipath streaming and set the video length to $10,000$ seconds. The startup delay ranges from 4 to 10 seconds.

In each setting, we repeat the simulation 30 times using independent random seeds. Let f_m and f_s denote the average fraction of late packets from the model and the simulation, respectively. We say that the model and the simulation match well if f_m falls within the 95% confidence interval obtained from the simulation, or $0.1 < f_m/f_s < 10$. The reason for the second condition is as follows. We classify a viewing quality as either satisfactory or unsatisfactory since our ultimate goal is to determine the conditions under which DMP-streaming provides satisfactory performance. In particular, we say a performance is satisfactory when the fraction of late packets from the model, f_m, is below 10^{-4} (see Section 7). Under this classification criteria and the condition that $0.1 < f_m/f_s < 10$, a good performance predicted by our model implies that the fraction of late packets is below 10^{-3} even in the worst case, which still corresponds to good performance (existing studies [Verscheure et al. 1998; Zhang et al. 2001] show that the viewing quality is not much degraded when the loss rate is below 10^{-3} to 10^{-4}).[3]

In the following, we first consider independent paths, that is, the multiple paths used by the video streams do not share a bottleneck link. We then consider correlated paths where the multiple paths share bottlenecks.

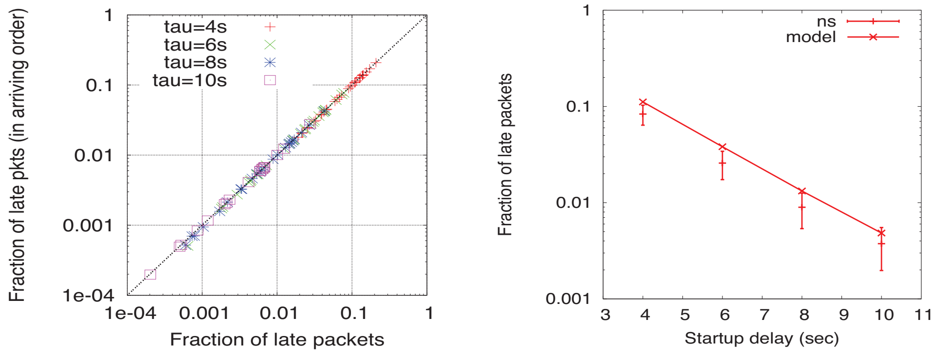
5.2 Independent Paths

For independent paths, we use the topology shown in Figure 3 (in Section 3). On path k, link (r_{k1}, r_{k2}) is the bottleneck link, where packets are lost due to buffer overflow, $k = 1, 2$. We validate our model

[3]On the other hand, a viewing quality may be good even when our model does not predict so. Therefore, our model provides a conservative prediction on whether a viewing quality is satisfactory or not.

Table II. Model Validation for Independent Paths (including both homogeneous and heterogeneous paths) in *ns*

Setting	p_1	p_2	R_1 (ms)	R_2 (ms)	T_{O_1}	T_{O_2}	μ (pkts ps)
1-1	0.023	0.023	210	210	1.6	1.6	50
2-2	0.037	0.036	150	150	1.7	1.7	50
3-3	0.053	0.053	200	200	1.9	1.9	30
4-4	0.034	0.035	80	80	3.0	3.3	80
1-2	0.023	0.036	210	150	1.6	1.7	50
1-3	0.023	0.053	210	200	1.6	1.9	40
2-3	0.036	0.053	150	200	1.7	1.9	40
3-4	0.049	0.032	200	80	1.9	3.0	60



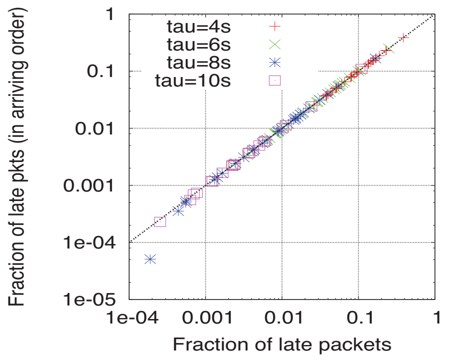
(a) Validating assumption in the model. (b) Fraction of late packets from *ns* and model.

Fig. 5. Validation results for independent homogeneous paths (Setting 2-2).

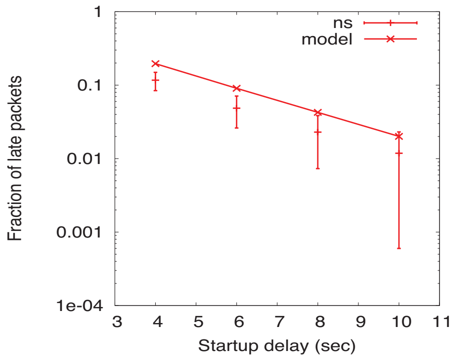
for several combinations of bottleneck link configurations that are shown in Table I. In particular, we consider homogeneous paths where the two paths use the same configuration, and heterogeneous paths where the two paths use different configurations.

5.2.1 *Independent Homogeneous Paths.* We consider four different settings with homogeneous paths, one for each configuration of the bottleneck link in Table I. When both paths use configuration i, we denote the setting as Setting i-i, $i = 1, \ldots, 4$. The parameters of the video streams are averaged over 30 simulation runs, as listed in Table II. The playback rate of the video is 30, 50, or 80 packets per second and each packet is 1500 bytes. Therefore, the bandwidth of the video is 360, 600, or 960Kbps.

In each setting, we validate that playing back packets in their arriving order leads to a similar fraction of late packets as when playing them back according to their playback times. This is to validate the simplifying assumption in our model (i.e., playing back packets in their arrival order; see Section 4.1). We also compare the fraction of late packets from our model and the simulation. Due to space limits, we only present the results for Setting 2-2; the results for other settings are similar [Wang et al. 2006]. Figure 5(a) plots the fraction of late packets when consuming packets in their arrival order versus that according to their playback times. We observe a close match and thus validate the simplifying assumption in our model. Figure 5(b) depicts the fraction of late packets from the model and the simulations (using the actual fraction of late packets). The 95% confidence intervals are from 30 simulation runs. We observe a good match between the model and the simulation.



(a) Validating assumption in the model.



(b) Fraction of late packets from *ns* and model.

Fig. 6. Validation results for independent heterogeneous paths (Setting 1-2).

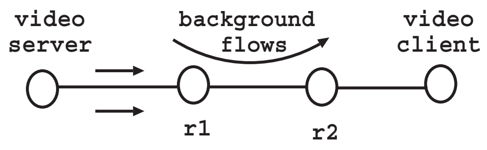


Fig. 7. Validation setting for correlated paths: The video server sends packets using two TCP flows on the same path. Packet losses are caused by buffer overflows on link (r_1, r_2).

Table III. Model Validation for Correlated Paths in *ns*

Setting	p_1	p_2	R_1 (ms)	R_2 (ms)	T_{O_1}	T_{O_2}	μ (pkts ps)
1	0.022	0.022	210	210	1.6	1.6	50
2	0.037	0.037	150	150	1.7	1.7	50
3	0.053	0.053	200	200	1.9	1.9	30
4	0.036	0.036	80	80	3.0	3.3	80

5.2.2 *Independent Heterogeneous Paths.* We consider four different settings with heterogeneous paths by pairing two different configurations for the bottleneck links listed in Table I. When the two paths use configuration i and j, we denote the setting as Setting i-j, $i, j = 1, \ldots, 4, i \neq j$. The parameters of the video streams are listed in Table II. The playback rate of the video is either 40, 50, or 60 packets per second. We next present the results for Setting 1-2; results for other settings are similar [Wang et al. 2006]. Figure 6(a) plots the fraction of late packets when consuming packets in their arrival order versus that according to their playback times. We observe a close match in all but one case (this mismatch might be due to an insufficient number of samples). Figure 6(b) depicts the fraction of late packets from the model and the simulation, again showing a good match.

5.3 Correlated Paths

For correlated paths, we consider an extreme case, namely, the video flows share the same path. The topology is show in Figure 7, where link (r_1, r_2) is the bottleneck link traversed by the video flows and background flows. We consider four settings, each with the bottleneck link configured using a configuration listed in Table I. If a setting uses configuration i, we refer to it as Setting i, $i = 1, \ldots, 4$. The parameters of the video streams are averaged over 30 simulation runs, as listed in Table III. As expected, the parameters of the two TCP streams are similar. The validation results are similar to those for independent homogeneous paths in Section 5.2.1 (figures omitted and are found in Wang et al. [2006]). This demonstrates that our model is also applicable to correlated paths as long as the loss processes of the two paths can be regarded as independent.[4]

6. MODEL VALIDATION USING EXPERIMENTS OVER THE INTERNET

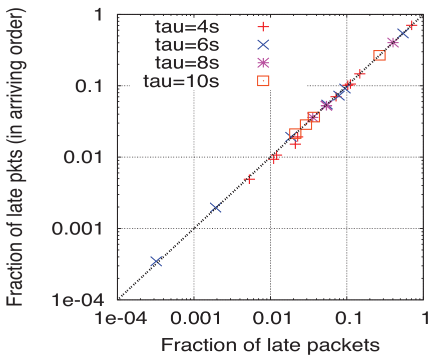
We have implemented DMP-streaming and validated our model for DMP-streaming through experiments conducted over the Internet. In each experiment, we use *tcpdump* to capture the packet timestamps on each path. The average loss rate, average RTT, and timeout value of each TCP flow are estimated from the *tcpdump* traces. We use Linux-based machines for all experiments.

Having no access to multihomed machines, we emulate multipath streaming by streaming from a server to two clients and then combining the traces at the two clients.[5] The video server is placed inside the University of Connecticut. The clients are chosen from nodes in Planetlab (http://www.planet-lab.org/). We use both homogeneous and heterogeneous paths. A setting with two homogeneous paths is created by streaming from the server to two nodes that are connected through ADSL in San Francisco, California. A setting with heterogenous paths is created by streaming from the server to one node in San Francisco, California and another in Hefei, China. The playback rate of the video is 25 or 50 packets per second when the paths are homogeneous and 100 packets per second when they are heterogenous. Each packet consists of 1448 bytes. Therefore, the bandwidth of the video varies from 300Kbps to 1.2Mbps. We conducted 10 experiments from July 21 to July 27, 2006 at randomly chosen times; each experiment lasted for 3,000 seconds.

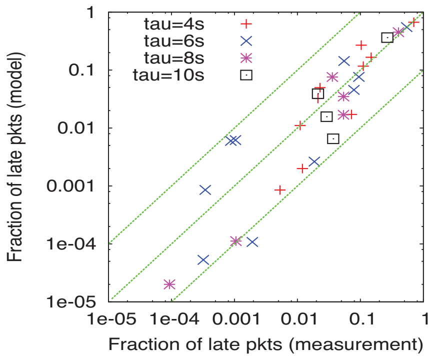
Figure 8(a) plots the fraction of late packets when playing back packets in their arriving order and according to their playback times. We again see that they are very close, and hence validate the assumption in our model. Figure 8(b) presents a scatterplot showing the fraction of late packets obtained from the measurements versus that predicted by the model. The 45-degree line starting at the origin represents a hypothetical perfect match between the measurements and the model. Along the upper and lower 45-degree lines, the fraction of late packets from the model is, respectively, 10 times higher and lower than that from the measurements. All but one scatterplot point fall within the upper and lower 45-degree lines, indicating a match between the model and the Internet experiments. When the startup delay is 10 seconds, in 6 experiments, the fraction of late packets is 0 (therefore these are not shown in the plot), and our model predictions are also 0 or close to 0 (the prediction is 0 for 5 experiments and 10^{-4} for one experiment). We speculate that the single discrepancy in Figure 8(b) between the model and the experimental results is due to an insufficient number of samples in the data trace.

[4]We validate that the loss processes of the two paths can be regarded as independent as follows. For each path, we obtain a sequence of average loss rate, one average in every 0.5 second. We then calculate the correlation coefficient of the two sequences from the two paths. The correlation coefficient of all the runs (over all the settings) is between 0.06 and 0.16, indicating little correlation between the loss processes in these two paths. This little correlation might be because packets from background flows are interspersed among the packets of the two TCP streams.

[5]Since the clients have different clocks, we synchronize the traces from the clients as follows. Before the streaming starts, the server sends a signal message to the two clients. The arrival time of the signal message at a client is regarded as a start time. We then subtract the start time from all subsequent timestamps at the client to obtain relative arrival times of the packets. Afterward, we combine the relative arrival times at the two clients to construct the entire arrival sequence. This way of synchronizing the packet traces ignores the difference in one-way delays from the server to the clients. However, since this difference is in milliseconds while the startup delay is in several seconds, we expect that its impact is negligible.



(a) validating assumption in the model



(b) results from experiments and model

Fig. 8. Model validation using experiments over the Internet.

7. EXPLORING THE PARAMETER SPACE

The primary goal of this section is to identify conditions under which DMP-streaming leads to a satisfactory performance. We say a performance is *satisfactory* when the fraction of late packets is less than 10^{-4} for a startup delay of around 10 seconds. This is because people can usually tolerate several seconds of startup delay and studies show that the viewing quality is not much degraded when the loss rate is below 10^{-3} to 10^{-4} (depending on quantization parameter) [Verscheure et al. 1998; Zhang et al. 2001].

We achieve the previous goal by varying the parameters in the model for DMP-streaming. The loss rate is varied from 0.004 to 0.04. The timeout value is varied from 1 to 4 times of the round-trip time, based on several measurements in Padhye et al. [1998], and our measurements in Section 6 and Wang et al. [2004]. The RTT is in a wide range of 40 ms to 300 ms. Let σ_a denote the *aggregate* achievable TCP throughput (in packets per second) over all the paths. Then $\sigma_a = \sum_{k=1}^{K} \sigma_k$, where σ_k is the achievable TCP throughput on the kth path. Throughout this section, we use $K = 2$; performance study under a larger number of paths is left as future work.

In the following, we first consider homogeneous paths, and then explore the impact of path heterogeneity. At the end, we discuss the benefits from DMP-streaming compared to single-path TCP-based streaming.

7.1 Conditions for Satisfactory Performance: Homogeneous Paths

Intuitively, the performance of multipath TCP streaming improves as the ratio of the aggregate achievable TCP throughput over the playback rate, σ_a/μ, increases. This is because, as σ_a/μ increases, packets accumulate in the client's local buffer faster relative to the playback rate of the video. We next vary the value of σ_a/μ from 1.2 to 2.0 to identify the minimum value of σ_a/μ that leads to satisfactory performance.

Since the paths are homogeneous, for ease of notation, we drop the index in the subscript and use p, R, T_O, and σ to denote, respectively, the loss rate, RTT, time-out value, and achievable TCP throughput on all the paths. Let σ_R denote the throughput on a path in one RTT. Then $\sigma_R = \sigma R$, and it is determined by p and T_O. Because $\sigma_a/\mu = K\sigma/\mu = K\sigma_R/(\mu R)$, we vary the value of σ_a/μ in one of the following two manners: (1) fixing σ_R (by fixing p and T_O) and μ, and varying the RTT R; and (2) fixing σ_R (by fixing p and T_O) and R, and varying the playback rate μ.

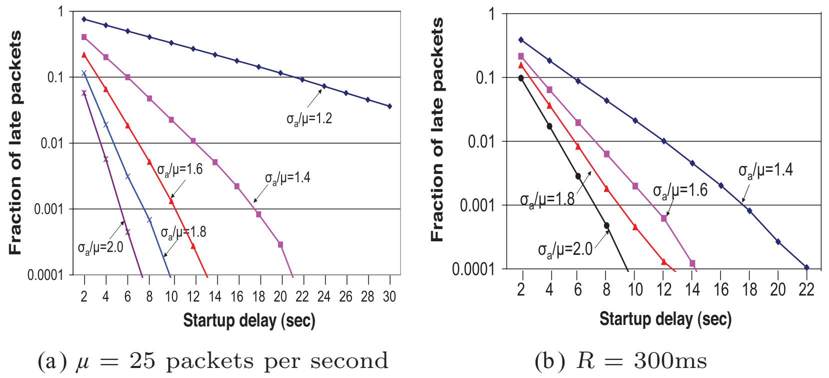


Fig. 9. Homogeneous paths: diminishing gain from increasing σ_a/μ on performance, $p = 0.02$, $T_O = 4$.

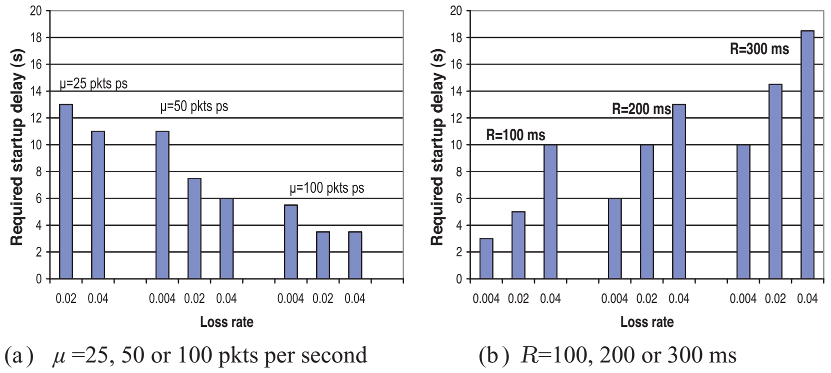


Fig. 10. Homogeneous paths: required startup delay so that the fraction of late packets is below 10^{-4}, $T_O = 4$, $\sigma_a/\mu = 1.6$.

We first present the results when varying σ_a/μ from 1.2 to 2.0 by varying the RTT R. The loss rate p is set to $0.004, 0.02$, or 0.04. The timeout value T_O is set to $1, 2, 3$, or 4. The playback rate μ is set to 25, 50, or 100 packets per second. We observe a diminishing gain from increasing σ_a/μ on performance. One example is shown in Figure 9(a), where $p = 0.02$, $T_O = 4$, and $\mu = 25$ packets per second. As shown in this figure, the performance improves dramatically as σ_a/μ increases from 1.2 to 1.4 and less dramatically afterwards. Figure 10(a) plots the required startup delays so that the fraction of late packets is below 10^{-4} when $T_O = 4$ (the required startup delays for lower T_O's are lower) and $\sigma_a/\mu = 1.6$. (The result for $p = 0.004$ and $\mu = 25$ packets per second is omitted because its corresponding RTT is over 600 ms, too large to correspond to a practical setting). We observe that the required startup delay is around or much below 10 seconds for all the settings. This indicates that the performance of DMP-streaming is satisfactory when σ_a/μ becomes 1.6.

We now consider the case where we vary σ_a/μ from 1.2 to 2.0 by varying the playback rate μ. Again, the loss rate p is set to $0.004, 0.02$, or 0.04; the timeout value T_O is set to $1, 2, 3$, or 4. The RTT R is set to 100, 200, or 300 ms. We again observe a diminishing gain from increasing σ_a/μ on performance. One example is shown in Figure 9(b), where $p = 0.02$, $T_O = 4$, and $R = 300$ ms. Figure 10(b) plots the required startup delays so that the fraction of late packets lies below 10^{-4} when $T_O = 4$ (the required startup delays for lower T_O's are lower) and $\sigma_a/\mu = 1.6$. It shows that the required startup delays are around or much below 10 seconds except for settings with a large RTT, high loss rate, and timeout value

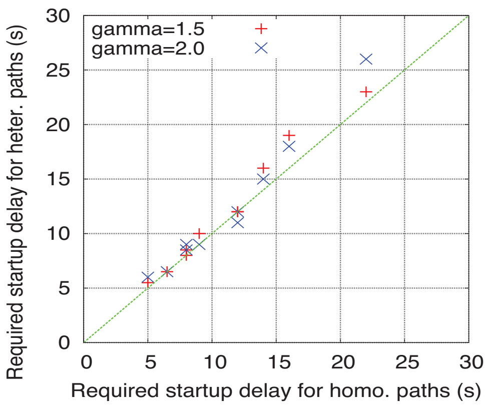


Fig. 11. Impact of path heterogeneity.

(e.g., $R = 300$ ms, $p = 0.04$, $T_O = 4$). For those settings, a higher σ_a/μ (e.g., $\sigma_a/\mu = 1.8$) is required to achieve a satisfactory performance.

7.2 Conditions for Satisfactory Performance: Heterogenous Paths

We study the impact of path heterogeneity on the performance of DMP-streaming. Once we understand the impact, we can relate the conditions for homogenous paths to those for heterogenous paths. More specifically, we compare the performance of DMP-streaming under two scenarios. In the first scenario, the two paths are homogeneous. Let p^o, R^o, T_O^o, and σ^o denote, respectively, the loss rate, RTT, timeout value, and the achievable TCP throughput on these two paths. In the second scenario, the paths are heterogeneous. Let p_k^e, R_k^e, $T_{O_k}^e$, and σ_k^e denote respectively the loss rate, RTT, timeout value, and the achievable TCP throughput on the kth path, $k = 1, 2$. We assume that a video with a playback rate μ is streamed in both scenarios. To make a fair comparison, we require that these two scenarios have the same aggregate achievable TCP throughput, namely, $\sum_{k=1}^{K} \sigma_k^e = K\sigma^o$. To make the exploration tractable, we use two methods to create paths with heterogeneous bitrate, one through heterogeneous RTTs, and the other through heterogeneous loss rates. These two methods are listed next, where γ will be referred to as the *heterogeneity factor*:

—*Case* 1. The two paths only differ in their RTTs. Specifically, $p_1^e = p_2^e = p^o$, $T_{O_1}^e = T_{O_2}^e = T_O^o$, $R_1^e = \gamma R^o$, $R_2^e = R^o/(2 - 1/\gamma)$, $\gamma > 1$. In this case, the aggregate achievable TCP throughputs under heterogeneous and homogeneous paths are the same since $\sigma_1^e + \sigma_2^e = \sigma^o(1/\gamma + 2 - 1/\gamma) = 2\sigma^o$.

—*Case* 2. The two paths only differ in loss rates, that is, $R_1^e = R_2^e = R^o$, $T_{O_1}^e = T_{O_2}^e = T_O^o$, $p_1^e = \gamma p^o$, $\gamma > 1$, and p_2^e is set using the formula for the achievable TCP throughput in Padhye et al. [1998] to obtain the same aggregate achievable TCP throughput as that under homogeneous paths.

We now report the results for the preceding two cases. All the settings to follow use $T_{O_1}^e = T_{O_2}^e = T_O^o = 4$. The heterogeneity factor γ is set to 2 or 1.5. In Case 1, we consider two loss rate settings, $p_1^e = p_2^e = p^o = 0.01$ or 0.04, representing relatively low and high loss rates. For homogeneous paths, $R^o = 150$ ms. For heterogeneous paths, when $\gamma = 2$, $R_1^e = 300$ ms and $R_2^e = 100$ ms; when $\gamma = 1.5$, $R_1^e = 225$ ms and $R_2^e = 112.5$ ms. In Case 2, we consider two RTT settings, $R_1^e = R_2^e = R^o = 100$ ms or 300 ms, representing relatively low and high RTTs. For homogeneous paths, $p^o = 0.02$. For heterogeneous paths, when $\gamma = 2$, $p_1^e = 0.04$ and $p_2^e = 0.012$; when $\gamma = 1.5$, $p_1^e = 0.03$ and $p_2^e = 0.014$. For each setting of the TCP parameters, the playback rate μ is set so that σ_a/μ=1.4, 1.6, or 1.8. We therefore have $(4 + 4) \times 3 = 24$ different settings for heterogeneous paths. Figure 11 plots the required startup delay

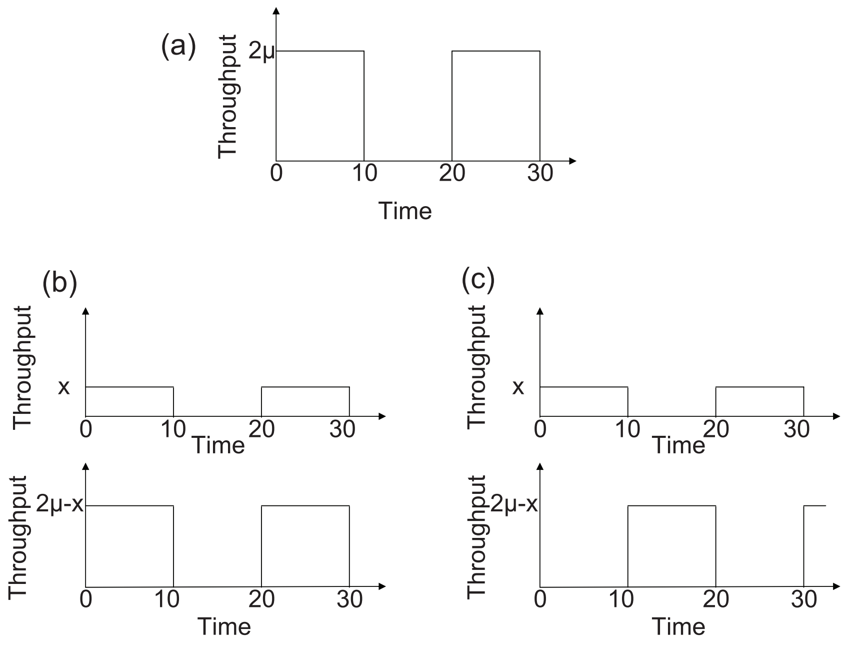


Fig. 12. Setting of a simple example to gain insights into the benefits from using DMP-streaming over single-path TCP-based live streaming.

(so that the late loss rate is below 10^{-4}) under homogeneous paths versus that under heterogeneous paths. We observe a close performance under homogeneous and heterogenous paths for all the settings. This indicates that the performance of DMP-streaming is not sensitive to path heterogeneity.

To obtain additional insights on the impact of path heterogeneity, we consider an extreme case where the achievable TCP throughput on one path is close to zero (e.g., when its loss rate is close to 1). In this extreme case, DMP-streaming sends most of the packets on the other path and becomes essentially single-path streaming. This extreme path-heterogeneity degrades the performance of DMP-streaming since it requires a higher σ_a/μ to achieve a satisfactory performance (single-path streaming generally requires $\sigma_a/\mu = 2$ for a satisfactory performance). However, when the path heterogeneity is less severe, we expect less impact from path heterogeneity as suggested by the results earlier.

7.3 Discussion: DMP-Streaming versus Single-Path TCP-Based Live Streaming

We have observed that DMP-streaming generally achieves satisfactory performance when the aggregate achievable TCP throughput is 1.6 times the video bitrate, with a few seconds of startup delay. This requirement is less than what is required in single-path TCP-based streaming (which generally requires that the achievable TCP throughput to be twice as the video bitrate [Wang et al. 2004]). The reason for this decrease is that DMP-streaming dynamically allocates packets onto the multiple paths to take advantage of path diversity provided by multipath. We next illustrate this using a simple example.

Suppose single-path streaming uses a single path P and DMP-streaming uses two paths P_1 and P_2. All paths periodically alternate between a zero and nonzero achievable TCP throughput, with the period of 10 seconds. The nonzero throughput on path P is 2μ packets per second, as shown in Figure 12(a). The nonzero throughputs on paths P_1 and P_2 are x and $(2\mu - x)$ packets per second, respectively, $x \in (0, \mu]$. Therefore, the aggregate throughput over paths P_1 and P_2 is μ packets per

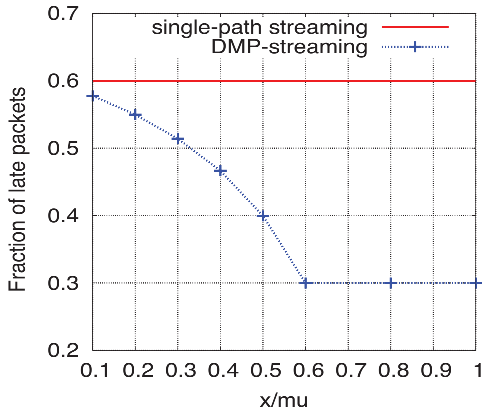


Fig. 13. DMP-streaming versus single-path TCP-based streaming in the simple example, $\tau = 4$ seconds.

second, equal to the average throughput of path P. We assume two cases for paths P_1 and P_2: (i) They experience congestion simultaneously, as shown in Figure 12(b); and (ii) they alternatively experience congestion, as shown in Figure 12(c). We further assume that the aforestated two cases each happen with a probability of 0.5. For both cases, we obtain the fraction of late packets; their average is the fraction of late packets under DMP-streaming.

We vary the startup delay, τ, from 1 to 9 seconds, and vary x from 0.1μ to μ. For all the settings, we observe that DMP-streaming leads to a lower fraction of late packets than single-path streaming. Figure 13 plots the results when the startup delay is 4 seconds (the results under other startup delays have a similar trend). We observe that the fraction of late packets under DMP-streaming decreases as x increases from 0.1μ to μ. Furthermore, for all values of x, DMP-streaming outperforms single-path streaming. The reason for the superior performance of DMP-streaming can be explained as follows. When paths P_1 and P_2 experience congestion simultaneously, the fraction of late packets under DMP-streaming equals that under single-path live streaming. However, when these two paths alternate to experience congestion, DMP-streaming intelligently sends packets to the uncongested path when one path is congested, and hence leads to a lower fraction of late packets.

8. COMPARISON WITH A STATIC SCHEME

We now compare DMP-streaming and a static streaming scheme to demonstrate the benefit of dynamic packet allocation in TCP-based multipath streaming. The static streaming scheme we consider is as follows. It assumes that the average available bandwidth on each path is known beforehand (e.g., through measurement), and allocates packets statically onto the multiple paths according to the available bandwidths. More specifically, the fraction of packets allocated to a path is proportional to the available bandwidth on that path. We refer to this static streaming scheme as *static-streaming*. In this scheme, the streaming process on a path is independent of those on other paths, and hence can be regarded as a separate single-path live streaming process. The fraction of late packets of the entire video can be obtained as follows. Consider a video with a playback rate of μ packets per second. Let α_i be the fraction of packets allocated onto the ith path. Let f_i be the fraction of late packets on the ith path. We obtain f_i using the model for single-path live streaming in Wang et al. [2004] where the packet generation rate and the playback rate are both $\alpha_i \mu$ packets per second. The fraction of late packets of the entire video is $\sum_{i=1}^{K} \alpha_i f_i$.

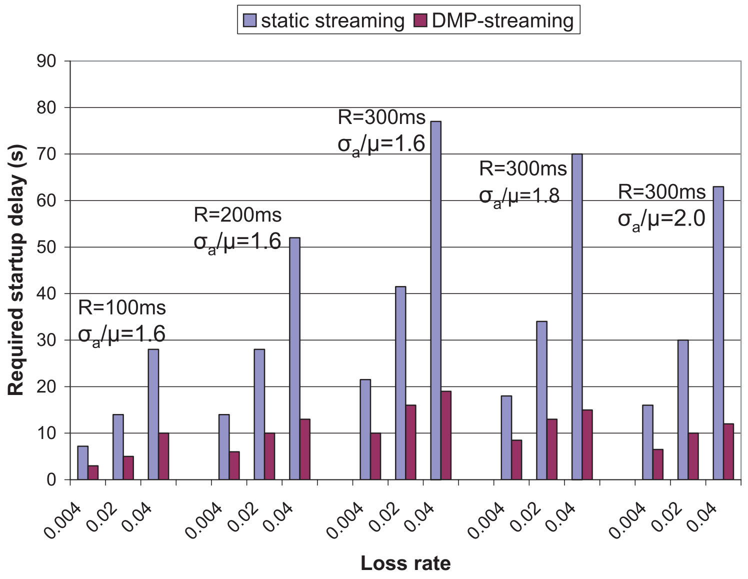


Fig. 14. Performance comparison of DMP-streaming and static streaming, $T_O = 4$.

Intuitively, DMP-streaming outperforms static-streaming since the allocation under DMP-streaming is dynamic, according to the *current* network bandwidths, while the allocation under static-streaming is static, *oblivious* to the current bandwidth changes. We next demonstrate quantitatively that DMP-streaming indeed outperforms static-streaming.

For simplicity, we consider two homogeneous paths, and let p, R, and T_O denote, respectively, the loss rate, RTT, and timeout value on both paths. Suppose a video, with playback rate μ, is to be streamed over these two paths. In this case, static-streaming assigns an equal number of packets to the two paths. Without loss of generality, we assume that it assigns odd numbered packets to the first path and even numbered packets to the second path. The loss rate p is set to 0.004, 0.02, or 0.04. The RTT R is 100, 200, or 300 ms. The timeout value T_O is 4. The video playback rate is varied such that σ_a/μ varies from 1.6 to 2. Figure 14 plots the results for several representative settings. We observe that, for the same setting, DMP-streaming significantly outperforms static-streaming: It requires a much lower startup delay for the fraction of late packets to be below 10^{-4}. This demonstrates the importance of dynamic packet allocation in multipath streaming.

To gain additional insights, let us consider the simple example from Section 7.3. In this example, the throughputs of two paths, P_1 and P_2, are under two cases (see Figures 12(b) and 12 (c)). Each case occurs with a probability of 0.5. Since the average TCP throughputs on paths P_1 and P_2 are x and $2\mu - x$, respectively, when using static-streaming, $x/(2\mu)$ of the packets are allocated onto path P_1, and $(2\mu - x)/(2\mu)$ of the packets are allocated onto path P_2. We can regard the streaming processes on paths P_1 and P_2 as two separate single-path streaming processes with playback rates $x/2$ and $(\mu - x/2)$ packets per second, respectively. It is easy to see that the fractions of late packets on both paths, and hence the average fraction of late packets of the entire video, are all equal to that under single-path streaming on path P (see Figure 12(a) for the setting of path P). Hence, as shown in Section 7.3, static-streaming leads to a larger fraction of late packets than DMP-streaming for all values of $x \in (0, \mu]$ and startup delays from 1 to 9 seconds. The reason for the inferior performance of static-streaming is that,

Table IV. DMP-Streaming (definition of the transition rate and the number of successfully transmitted packets during a state transition for the kth TCP flow, $k = 1, \ldots, K$)

$r^k_{w,0,0,0,0;w,1,0,0}$	=	$(1 - p_k)^w / R_k$,	$1 \le w \le W_{max}$
$r^k_{w,1,0,0,0;w+1,0,0,0}$	=	$(1 - p_k)^w / R_k$,	$1 \le w \le W_{max}$
$r^k_{w,1,0,0,0;w,0,0,0}$	=	$(1 - p_k)^w / R_k$,	$w = W_{max}$
$s^k_{w,0,0,0,0;w,1,0,0}$	=	w,	$1 \le w \le W_{max}$
$s^k_{w,1,0,0,0;w+1,0,0,0}$	=	w,	$1 \le w \le W_{max}$
$s^k_{w,1,0,0,0;w,0,0,0}$	=	w,	$w = W_{max}$
$r^k_{w,c,0,0,0;w-l,0,l,0,0}$	=	$p_k(1 - p_k)^{w-l} / R_k$,	$2 \le w \le W_{max}$, $c = 0, 1, 1 \le l \le w$,
$r^k_{w,c,0,0,0;1,0,0,1,1}$	=	p_k / R^k_{TO},	$2 \le w \le W_{max}, c = 0, 1$
$s^k_{w,c,0,0,0;w-l,0,l,0,0}$	=	$w - l$,	$2 \le w \le W_{max}$, $c = 0, 1, 1 \le l \le w$,
$s^k_{w,c,0,0,0;1,0,0,1,1}$	=	0,	$2 \le w \le W_{max}, c = 0, 1$
$r^k_{1,0,l,0,0;1,0,0,1,1}$	=	$1/(R^k_{TO} - R_k)$,	
$r^k_{2,0,l,0,0;1,0,0,1,1}$	=	$1/(R^k_{TO} - R_k)$,	
$r^k_{w,0,l,0,0;1,0,0,1,1}$	=	$\sum_{i=1}^{2} p_k(1 - p_k)^i / (R^k_{TO} - R_k)$,	$3 \le w \le W_{max}$
$r^k_{w,0,l,0,0,n;\lfloor (w+l)/2 \rfloor,0,0,0,0,n'}$	=	$(\sum_{i=3}^{w-1} p_k(1 - p_k)^i + (1 - p_k)^w) / R_k$,	$3 \le w \le W_{max}$,
$s^k_{1,0,l,0,0;1,0,0,1,1}$	=	$1 - p_k$,	
$s^k_{2,0,l,0,0;1,0,0,1,1}$	=	$p_k(1 - p_k) + 2(1 - p_k)^2$,	
$s^k_{w,0,l,0,0;1,0,0,1,1}$	=	$\frac{\sum_{i=0}^{2} i p_k (1-p_k)^i}{\sum_{i=0}^{2} p_k (1-p_k)^i}$,	$1 \le w \le W_{max}$
$s^k_{w,0,l,0,0;\lfloor (w+l)/2 \rfloor,0,0,0,0}$	=	$\frac{\sum_{i=3}^{w-1} i p_k (1-p_k)^i + w(1-p_k)^w}{\sum_{i=3}^{w-1} p_k (1-p_k)^i + (1-p_k)^w}$,	$3 \le w \le W_{max}$
$r^k_{1,0,0,i,q;1,0,0,\min\{i+1,7\},1}$	=	$p_k / (2^{(i-1)} R^k_{TO})$,	$1 \le i \le 7, q = 0, 1$
$r^k_{1,0,0,i,1;1,0,0,i,0}$	=	$(1 - p_k)/R_k$,	$1 \le i \le 7$
$r^k_{1,0,0,i,0;2,0,0,0,0}$	=	$(1 - p_k)/R_k$,	$1 \le i \le 7$
$s^k_{1,0,0,i,q;1,0,0,\min\{i+1,7\},1}$	=	0,	$1 \le i \le 7, q = 0, 1$
$s^k_{1,0,0,i,1;1,0,0,i,0}$	=	1,	$1 \le i \le 7$
$s^k_{1,0,0,i,0,n;2,0,0,0,0,n+1}$	=	1,	$1 \le i \le 7$

since packets are allocated beforehand in a static manner, when one path is congested while the other is not, the packets allocated to the congested path cannot be moved to the uncongested path.

9. CONCLUSIONS AND FUTURE WORK

In this article, we designed DMP-streaming, a simple and practical scheme for multipath streaming via TCP. We further developed an analytical model for DMP-streaming and validated the model using extensive *ns* simulation and Internet experiments. Using this model, we explored the parameter space and found that the performance of DMP-streaming is not sensitive to path heterogeneity. Furthermore, the performance is generally satisfactory when the aggregate achievable TCP throughput is 1.6 times the video bitrate, with a few seconds of startup delay. Last, DMP-streaming significantly outperforms alternative static streaming schemes.

Our results also demonstrated the benefits of using multiple paths versus a single path for TCP-based multimedia streaming. In practice, a multipath can be obtained through multihoming, an increasingly common practice in enterprise networks. In residential networks, although home users typically subscribe to a single broadband provider, neighborhood networks may be a viable approach to provide users multiple points of access to the Internet. Throughout the article, we used the fraction of late packets as the performance metric; performance study using more complicated and user-oriented metrics is left as future work. We will also broaden our scope to consider Variable BitRate (VBR) videos, study performance trade-offs under a larger number of paths (i.e., more than two paths), and investigate the scalability of using multipaths for streaming from a server to a large number of clients.

APPENDIX

A. MODEL FOR DMP-STREAMING

Let the tuple $(X_1(t), \dots, X_K(t), N(t))$ represent the continuous-time Markov chain for DMP-streaming, where $X_k(t)$ is the state of the TCP flow on the kth path and $N(t)$ is the number of early packets in the client's local buffer at time t, $k = 1, \dots, K$, $t \geq 0$. The transition of the Markov chain occurs under two types of events: (1) when a TCP flow makes a transition, and (2) when a packet is consumed (played back) from the client's local buffer. The rate for the latter event is $1/\mu$, where μ is the playback rate of the video. We next describe the transition rates of the various TCP flows.

The state transitions of the different TCP flows are independent of each other. Consider the kth TCP flow. Its state at time t, $X_k(t)$, is a tuple, $X_k(t) = (W_k(t), C_k(t), L_k(t), E_k(t), Q_k(t))$, where $W_k(t)$ is the window size; $C_k(t)$ models the delayed ACK behavior of TCP ($C_k(t)$ is initially 0 and alternates between 0 and 1 after each state transition); $L_k(t)$ is the number of packets lost when the previous transition occurs; $E_k(t)$ denotes whether the connection is in a timeout state and the value of the back-off exponent; $Q_k(t)$ indicates whether a packet being sent in the timeout phase is a retransmission ($Q_k(t) = 1$) or a new packet ($Q_k(t) = 0$). Let $r^k_{w,c,l,e,q;w',c',l',e',q'}$ denote the transition rate from state (w, c, l, e, q) to state (w', c', l', e', q'). Let $s^k_{w,c,l,e,q;w',c',l',e',q'}$ denote the number of packets that are successfully transmitted during the transition from state (w, c, l, e, q) to state (w', c', l', e', q'). Let p_k and R_k denote, respectively, the loss probability and the RTT, and let R^k_{TO} denote the value of the first retransmission timer for the kth TCP flow. Denote the maximum window size as W_{max}. Then Table IV lists $r^k_{w,c,l,e,q;w',c',l',e',q'}$ and $s^k_{w,c,l,e,q;w',c',l',e',q'}$ for the kth TCP flow, $k = 1, \dots, K$. In the table, there are four groups of r's and s's corresponding respectively to situations (1) no packets are lost when a transition happens; (2) one or more packets are lost when a transition happens; (3) one or more packets are lost when the current transition happens and at least one packet was lost when the previous transition happened; and (4) exponential back-off.

ACKNOWLEDGMENTS

We thank Y. Gu for helpful discussion on simulation.

REFERENCES

Altman, E., Avrachenkov, K., and Barakat, C. 2000. A stochastic model of TCP/IP with stationary random losses. In *Proceedings of the ACM SIGCOMM*, 231–242.

Apostolopoulos, J., Wong, T., Tan, W., and Wee, S. 2002. On multiple description streaming with content delivery networks. In *Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*.

Bohacek, S. 2003. A stochastic model of TCP and fair video transmission. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*.

Cardwell, N., Savage, S., and Anderson, T. 2000. Modeling TCP latency. In *Annual Joint Conference of the Computer and Communications Societies (InfoCom) (3)*. 1742–1751.

CHESTERFIELD, J., CHAKRAVORTY, R., PRATT, I., BANERJEE, S., AND RODRIGUEZ, P. 2005. Exploiting diversity to enhance multimedia streaming over cellular links. In *Proceedings of the Annual Joint Conference of the IEEE Computer and Communications Societies (InfoCom)*.

DE CUETOS, P., GUILLOTEL, P., ROSS, K. W., AND THOREAU, D. 2002. Implementation of adaptive streaming of stored MPEG-4 FGS video over TCP. In *Proceedings of the International Conference on Multimedia and Expo*.

DE CUETOS, P. AND ROSS, K. W. 2002. Adaptive rate control for streaming stored fine-grained scalable video. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*.

DE SOUZA E SILVA, E. AND LEAO, R. M. M. 2000. The TANGRAM-II environment. In *Proceedings of the 11th International Conference on Modeling Tools and Techniques for Computer and Communication System Performance Evaluation*.

FIGUEIREDO, D. R., LIU, B., MISRA, V., AND TOWSLEY, D. 2002. On the autocorrelation structure of TCP traffic. *Comput. Netw. J*.

GOLUBCHIK, L., LUI, J., TUNG, T., CHOW, A., LEE, W., FRANCESCHINIS, G., AND ANGLANO, C. 2002. Multipath continuous media streaming: What are the benefits? *Perform. Eval.*

HSIEH, H.-Y. AND SIVAKUMAR, R. 2002. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. In *Proceedings of the ACM Annual International Conference on Mobile Computing and Networking (MobiCom)*.

JURCA, D. AND FROSSARD, P. 2006. Packet selection and scheduling for multipath video streaming. *IEEE Trans. Multimedia*.

KIM, T. AND AMMAR, M. 2006. Receiver buffer requirements for video streaming over TCP. In *Proceedings of the Visual Communications and Image Processing Conference*.

KRASIC, C. AND WALPOLE, J. 2001. Priority-progress streaming for quality-adaptive multimedia. In *Proceedings of the ACM Multimedia Doctoral Symposium*.

LI, M., CLAYPOOL, M., KINICKI, R., AND NICHOLS, J. 2003. Characteristics of streaming media stored on the Internet. Tech. rep. WPI-CS-TR-03-18, Computer Science Department, Worcester Polytechnic Institute. May.

LIANG, Y. J., STEINBACH, E. G., AND GIROD, B. 2001. Real-time voice communication over the Internet using packet path diversity. In *Proceedings of the ACM Multimedia Conference*.

MATHIS, M., SEMKE, J., AND MAHDAVI, J. 1997. The macroscopic behavior of the TCP congestion avoidance algorithm. *Comput. Commun. Rev. 27*, 3.

MELLIA, M., STOICA, I., AND ZHANG, H. 2002. TCP model for short lived flows. *IEEE Commun. Lett. 6*, 2.

NGUYEN, T. P. AND AVIDEH, Z. 2004. Mutiple sender distributed video streaming. *IEEE Trans. Multimedia 6*, 2, 315–326.

PADHYE, J., FIROIU, V., AND TOWSLEY, D. 1999. A stochastic model of TCP Reno congestion avoidance and control. Tech. rep. 99-02, Department of Computer Science, University of Massachusetts, Amherst.

PADHYE, J., FIROIU, V., TOWSLEY, D., AND KUROSE, J. 1998. Modeling TCP throughput: A simple model and its empirical validation. In *Proceedings of the ACM SIGCOMM*, 303–314.

REJAIE, R. AND ORTEGA, A. 2003. PALS: Peer-to-peer adaptive layered streaming. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*.

RIBEIRO, B., DE SOUZA E SILVA, E., AND TOWSLEY, D. 2005. On the efficiency of path diversity for continuous media applications. Tech. rep. 05-19, Department of Computer Science, University of Massachusetts, Amherst.

SEELAM, N., SETHI, P., AND CHI FENG, W. 2001. A hysteresis-based approach for quality, frame rate, and buffer management for video streaming using TCP. In *Proceedings of the Conference on Management of Multimedia Networks and Services*.

SHARMA, V., KALYANARAMAN, S., KAR, K., RAMAKRISHNAN, K., AND SUBRAMANIAN, V. 2008. MPLOT: A transport protocol exploiting multipath diversity using erasure codes. In *Proceedings of the IEEE Annual Joint Conference of the Computer and Communications Societies InfoCom*.

SRIPANIDKULCHAI, K., MAGGS, B., AND ZHANG, H. 2004. An analysis of live streaming workloads on the Internet. In *Proceedings of the Internet Measurement Conference (IMC)*. 41–54.

TULLIMAS, S., NGUYEN, T., EDGECOMB, R., AND CHEUNG, S. 2008. Multimedia streaming using multiple TCP connections. *ACM Trans. Multimedia Comput. Commun. Appl. 4*, 3.

VAN DER MERWE, J., SEN, S., AND KALMANEK, C. 2002. Streaming video traffic: Characterization and network impact. In *Proceedings of the International Web Content Caching and Distribution Workshop*.

VERSCHEURE, O., FROSSARD, P., AND HAMDI, M. 1998. MPEG-2 video services over packet networks: Joint effect of encoding rate and data loss on user-oriented QoS. In *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV)*.

WANG, B., KUROSE, J., SHENOY, P., AND TOWSLEY, D. 2004. Multimedia streaming via TCP: An analytic performance study. In *Proceedings of the ACM Multimedia*.

WANG, B., WEI, W., GUO, Z., AND TOWSLEY, D. 2006. Multipath live streaming via TCP: Scheme, performance and benefits. Tech. rep. BECAT/CSE-TR-06-7, Computer Science and Engineering Department, University of Connecticut.

Wang, B., Wei, W., Guo, Z., and Towsley, D. 2007. Multipath live streaming via TCP: Scheme, performance and benefits. In *Proceedings of the ACM CoNEXT Conference*.

Wang, Y., Claypool, M., and Zuo, Z. 2001. An empirical study of video performance across the Internet. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*.

Zhang, Q., Zhu, W., and Zhang, Y.-Q. 2001. Resource allocation for multimedia streaming over the Internet. *IEEE Trans. Multimedia*. *3*, 3.

Received February 2008; revised June 2008; accepted August 2008