Multipath Live Streaming via TCP: Scheme, Performance and Benefits*

Bing Wang
University of Connecticut

Wei Wei
United Technologies Research Center

Zheng Guo
University of Connecticut

Don Towsley
University of Massachusetts, Amherst

ABSTRACT

Motivated by the wide use of TCP for streaming in practice and the increasing availability of multipath between end hosts, we study multipath live streaming via TCP in this paper. 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, with a few seconds of startup delay. Last, we comment on the benefits of using multipath versus single path for TCP-based streaming.

1. INTRODUCTION

The increasing availability of multipath between end hosts has motivated a number of recent studies on multipath audio/video streaming (i.e., streaming over multiple network paths) [11, 3, 19, 22, 26]. 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 realtime requirement of multimedia streaming.

In this paper, 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 [31]. 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 [33, 28, 29]. In our earlier work [31], 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, with a few seconds of startup delay. This result partly explains why TCP streaming has been an attractive option in practice: such a bandwidth requirement can be satisfied even for broadband home users (using cable modem or ADSL technologies) for a large fraction of streaming multimedia clips in the Internet today.

Motivated by the above observations, we focus on multipath live streaming via TCP. 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, compared to using a single path, in TCP-based live streaming?*

Our paper answers the above questions and makes the following main contributions:

- We design a simple and practical scheme, named *Dynamic MPath-streaming (DMP-streaming)*, for multipath streaming via TCP. It dynamically dis-

*This work was supported in part by UCONN large grant, and the National Science Foundation under ANI-0325868, ANI-0240487, ANI-0085848, and EIA-0080119.

tributes packets over the multiple paths (to accommodate bandwidth fluctuations) by *implicitly inferring* the available bandwidths on these paths.

- We develop an analytic model for DMP-streaming. This model allows a *systematic* performance study, a task that is difficult when using empirical measurements or simulation alone. We validate this model using extensive *ns* simulation and Internet experiments. 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, 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.

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 above 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 [31]. 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. Hence, in addition to economical reasons (subscribing to multiple low-bandwidth access links is cheaper than subscribing to a single high-bandwidth access link [12]), it is also advantageous to use multipath for TCP-based streaming due to performance reasons.

The rest of the paper 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* simulations and Internet experiments, respectively. Section 7 explores the parameter space of the model. Finally, Section 8 concludes the paper.

1.1 Related work

Multipath continuous media streaming is studied in [11, 3, 19, 22, 26, 15, 6]. In particular, [11, 26] demonstrate the benefits of using multiple paths for continuous media streaming. In [3, 19], coding techniques (Multiple Description Codes) are applied to the video streams to improve loss recovery. The study in [22] determines the sending rate and the distribution of packets on the multiple paths to minimize loss rate at the receiver. The work in [15] proposes a heuristic algorithm for multipath video streaming that provides close to optimal performance. The study in [6] proposes a multipath streaming scheme suitable for cellular links. All the above studies use UDP as the transport protocol. We study multipath streaming via TCP, which is fundamentally different from UDP-based streaming (e.g., a UDP-based streaming might not use error recovery and/or congestion control mechanisms; even if it uses, the mechanisms are very different from those in TCP). To the best of our knowledge, our work is the first study on multipath streaming via TCP. Our performance study focuses on wired networks (although DMP-streaming can be applied to wireless networks).

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 [27, 17, 8, 7] focus on how to adapt the video bitrate to deal with network bandwidth fluctuations. Our earlier work [31] studied the performance of multimedia streaming using TCP. A recent study [16] analytically determines the proper receiver buffer size to ensure a prescribed video quality for TCP streaming. All the above studies use one TCP flow on a single path, while we use multiple TCP flows in this paper.

The literature on TCP modeling is extensive. Much of the TCP modeling focuses on TCP performance for file transfers, assuming long-lived flows [20, 24, 23, 2, 10] or short-lived flows [5, 21]. The study of [4] models TCP congestion window to determine a TCP-friendly transmission rate for UDP video flows. Our study differs from the above in that we develop analytical models for multipath TCP streaming, with a focus on multipath and the timeliness of the packets.

Last, [13] points 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 wired network (which has relatively stable bandwidth) for multimedia streaming (which can tolerate 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 (a special case is that they are the same path). 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. Each packet is associated with a packet number (corresponding to the position, and hence the playback time, of the packet) so that packets from the multiple paths can be reassembled at the receiver. The client allows a startup delay, τ, that is on the order of seconds, a common practice in commercial streaming products. All packets arriving earlier than their playback times are stored in the client's local buffer, which is assumed 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.

We consider a constant bit rate (CBR) video, motivated from measurement results that most videos streamed over the Internet are CBR [18]. 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*. A late packet typically leads to a glitch during playback. Therefore, we use the *fraction of late packets*, i.e., 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 characteristics of live streaming. The first is on the number of *early packets* (i.e., packets arriving earlier than their playback times); the second is on the TCP throughputs on the multiple paths.

2.1 Number of early packets in live streaming

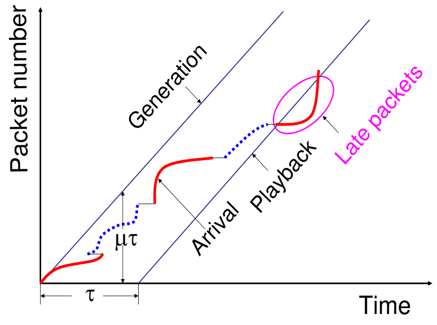


Figure 1: Illustration of multipath live streaming via TCP. In this example, $K = 2$, solid and dashed curves represent packet arrivals from the first and the second path respectively.

Fig. 1 illustrates multipath live streaming via TCP. The video server generates packets at the constant rate equal to the playback rate of the video (i.e., μ packets per second). For ease of exposition, we 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 $G(t) = \mu t$. At the client side, let $B(t)$ denote the number of packets played back by the client by time t. Then $B(t) = \mu(t - \tau)$, $t \geq \tau$. 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 $G(t) - B(t) = \mu t$ at any time t. This characteristic is to be 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 backlogged TCP source, i.e., a source always having 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. SCHEME FOR MULTIPATH STREAMING VIA TCP

In live-streaming, 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

```
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)
      }
   }
```

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

occasionally lower than the video playback rate; buffering as many packets as possible can compensate this adverse effect. 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 their current network bandwidths and allocates more packets on paths with higher throughputs (as in existing UDP-based schemes, e.g., [25]). Furthermore, it should avoid explicitly probing for bandwidth on each path (so no probing traffic is generated). We develop a TCP-based streaming scheme that satisfies the above desired properties and name it *Dynamic MPath-streaming (DMP-streaming).* We next describe this scheme in detail.

Our DMP-streaming scheme is summarized in Fig. 2. The server places the generated video packets into a queue, referred to as *server queue*. In the server queue, packets with earlier playback times are placed at the head of the queue. 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 obtains the 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 the its sending buffer before transmitting them into the network. The TCP sender cannot send any more packets (i.e., the TCP sender blocks) when the sending buffer is full. Therefore, once the TCP sending buffers on the multiple paths become 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 when the multiple TCP flows share or do not share bottleneck links (a special case is that they share the same path). Furthermore, it is also applicable to stored-video streaming. We focus on using DMP-streaming for live streaming in this paper; its performance for stored-video streaming is left as future work.

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 our model and then describe our model in detail.

4.1 Overview of the model

Our model assumes a single TCP flow on each path from the server to the 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 $N_{max} = \mu\tau$ (see Section 2.1).

One challenge in modeling multipath TCP streaming is that, although packets from each path arrive in order, packets from the multiple paths may arrive out of order. For instance, suppose that packet i is lost by a TCP flow and a later packet, packet j $(j > i)$, is sent successfully by another TCP flow. Then packet j may arrive at the client earlier than packet i and this leads to out-of-order packets at the receiver. 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, as confirmed by our simulation and experimental results (in Sections 5 and 6), out-of-order packets only have a negligible effect on the fraction of late packets in DMP-streaming. In other words, playing back packets in their arriving order only causes a negligible error. Therefore, in our model, we only keep track of the number of early packets and play back packets as if they were in order. The reasons why the effect of out-of-order packets is negligible 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. Then when playing back packets in their arriving order, packet j is played back as packet i. This case, however, does not cause an error to the fraction of late packets (since neither of packets i and 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 and this 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.

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 k-th 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. For the k-th TCP flow, its state at time t, $X_k(t)$, is a tuple containing five components, i.e., $X_k(t) = (W_k(t), C_k(t), L_k(t), E_k(t), Q_k(t))$, where $W_k(t)$ is the window size; $C_k(t)$ models the delayed ACK behavior of TCP ($C_k(t) = 0$ and $C_k(t) = 1$ indicate the first and the second of the two rounds respectively); $L_k(t)$ is the number of packets lost in the previous round; $E_k(t)$ denotes whether the connection is in a timeout state and the value of the back-off exponent; $Q_k(t)$ indicates if 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 state depends on these two states and the parameters of this TCP flow, including its RTT, loss rate and timeout value.[1] Due to space limits, a detailed description of the state transition rate for each TCP flow is found in [32].

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, (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 event that triggers the transition at time t. Let $\mathcal{E}(t) = \mathcal{P}$ denote that a transition of a TCP flow triggers the transition at time t. Similarly, let $\mathcal{E}(t) = \mathcal{C}$ denote that a packet consumption triggers the transition at time t. Then considering these two conditions,

- Condition 1 ($\mathcal{E}(t) = \mathcal{P}$): Suppose the k-th 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 k-th TCP flow transmits successfully at the transition (details in [32]).
- Condition 2 ($\mathcal{E}(t) = \mathcal{C}$): then the number of early packets, $N(t)$, is reduced by 1.

Note from the above 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 TANGRAM-II modeling tool [9].

The above 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

[1]Our assumption on loss process follows [23, 10]. That is, 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.

Config.	FTP flows	HTTP flows	Prop. Delay (ms)	B.w. (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

Table 1: Configurations of the bottleneck link.

our model when the TCP flows share and not share bottleneck links in Section 5.

5. MODEL VALIDATION USING *NS*

In this section, we validate our model for DMP-streaming using *ns*. We use $K = 2$, i.e., 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 at the link and the number of background flows traversing that link. These configurations are listed in Table 1.

For the video stream (via TCP) on the k-th path, let p_k, R_k and R_{TO}^k denote respectively the loss probability, the round-trip time (RTT), and the first retransmission timer. We further define $T_{O_k} = R_{TO}^k/R_k$; T_{O_k} 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 make 30 runs. 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 one of the following two conditions is satisfied: f_m falls within the confidence interval obtained from the simulation, or $0.1 < f_m/f_s < 10$. The reason for the second condition is as follows. When f_m and f_s lie within the same order of magnitude of each other, we regard that they correspond to similar viewing quality (the quality is classified as either satisfactory or unsatisfactory since our ultimate goal is to determine the conditions under which TCP provides satisfactory streaming performance).

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.

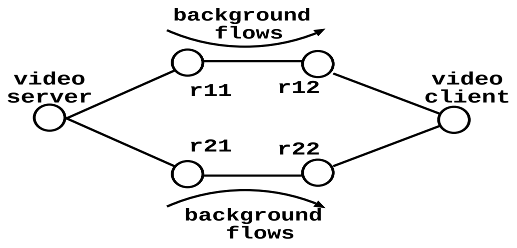


Figure 3: Validation setting for independent paths in *ns*: the video server spreads the video over two paths to the client. Packet losses are caused by buffer overflows on the bottleneck links from router r_{k1} to r_{k2}, $k = 1, 2$.

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

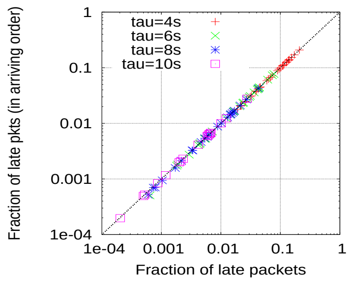
Table 2: Model validation for independent paths (including both homogeneous and heterogeneous paths) in *ns*.

5.2 Independent paths

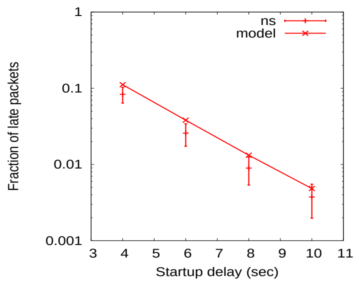
For independent paths, we use the topology shown in Fig. 3. A video server streams a video to a client via TCP over two paths. On path k, the link (r_{k1}, r_{k2}) is the bottleneck link, where packets are lost due to buffer overflow, $k = 1, 2$. The link from the video server to r_{k1} (and the link from r_{k2} to the video client) has propagation delay of 10 ms and bandwidth of 100 Mbps. We validate our model for several combinations of bottleneck link configurations shown in Table 1. 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 4 different settings with homogeneous paths, one for each configuration of the bottleneck link in Table 1. 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 2. 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

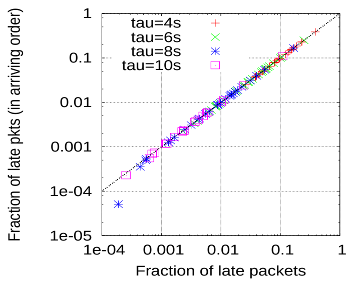


(a) Effect of out-of-order packets.

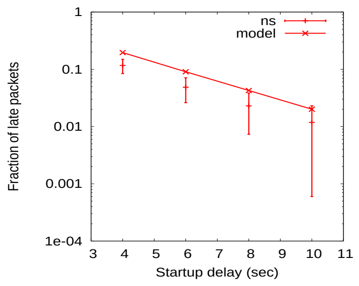


(b) Fraction of late packets from *ns* and model.

Figure 4: Validation results for independent homogeneous paths (Setting 2-2).



(a) Effect of out-of-order packets.



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

Figure 5: Validation results for independent heterogeneous paths (Setting 1-2).

video is 360, 600 or 960 Kbps.

In each setting, we validate that the effect of out-of-order packets can be ignored (an assumption in our model) and 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 [32]. We fist validate that the effect of out-of-order packets is negligible (see Section 4.1). From the simulation trace, we obtain the fraction of late packets when playing back packets in their arriving order and that according to their playback times. Fig. 4(a) is a scatter-plot of these two quantities. We observe a close match and thus validate that the effect of out-of-order packets is negligible. Fig. 4(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.

5.2.2 *Independent heterogeneous paths*

We consider 4 different settings with heterogeneous paths by pairing two different configurations for the bottleneck links listed in Table 1. 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 2. The playback rate of the video is either 40, 50 or 60 packets per second. We next only present the results for Setting 1-2; results for other settings are similar [32]. Fig. 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 for all the settings except one with a low fraction of late packets (this mismatch might be due to insufficient number of samples). This again validates that the effect of out-of-order packets can be ignored. Fig. 5(b) depicts the fraction of late packets from the model and the simulation. We again observe a good match between the model and the simulation results.

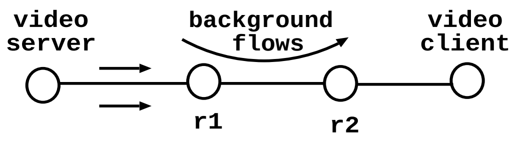
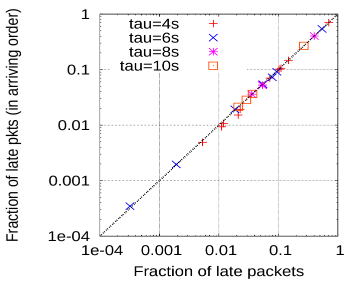


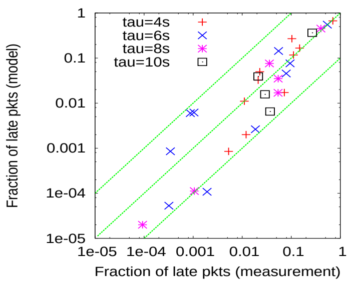
Figure 6: Validation setting for correlated paths: the video server sends packets using two TCP flows on the same path. Packet losses are caused by buffer overflows on the link (r_1, r_2).

5.3 Correlated paths

For the case of correlated paths, we consider an extreme case, namely, the video flows share the same path.



(a) Effect of out-of-order packets.



(b) Results from experiments and model.

Figure 7: Model validation using experiments over the Internet.

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

Table 3: Model validation for correlated paths in *ns*.

The topology is show in Fig. 6. The 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 1. 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 3. 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 due to space limits). This demonstrates that our model is also applicable to corrected paths as long as the loss processes of the two paths can be regarded as independent, conforming to the assumptions in our model. In the above settings, packets from background flows are interspersed among the packets of the two TCP streams, which reduces the correlation of the loss processes of these two TCP streams.

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. The video server is placed inside University of Connecticut. The clients are chosen from nodes in Planetlab [1]. 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 (under homogeneous paths) and 100 packets per second (under heterogenous paths). Each packet consists of 1448 bytes. Therefore, the bandwidth of the video varies from 300 Kbps to 1.2 Mbps. We conducted 10 experiments from July 21 to July 27, 2006 at randomly chosen times; each experiment lasted for $3,000$ seconds.

Fig. 7(a) plots the fraction of late packets when playing back packets in their arriving order and that according to their playback times. We again see that they are very close, and hence validate that the effect of out-of-order packets is negligible. Fig. 7(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 are not shown in the plot) while our model predictions are also 0 for 5 experiments and 10^{-4} for one experiment. We speculate that this single discrepancy between the model and the experimental results is due to an insufficient number of samples in the data trace.

7. EXPLORING THE PARAMETER SPACE

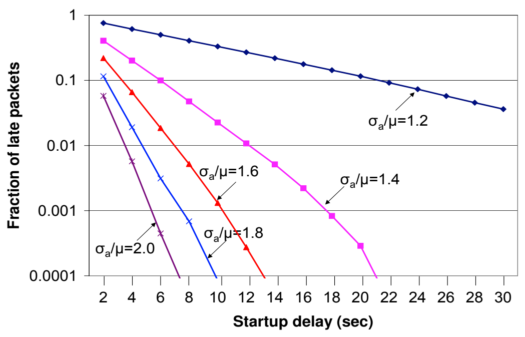


Figure 8: Diminishing gain from increasing σ_a/μ on performance, $p = 0.02$, $T_O = 4$, $\mu = 25$ packets per second.

In this section, we explore the impacts of the various parameters on the performance of DMP-streaming using the model developed in Section 4. Our goals are: (1) identify conditions under which DMP-streaming leads to a satisfactory performance; (2) compare DMP-streaming and a static scheme to quantify the benefits from dynamic packet allocation in TCP-based multipath streaming. We achieve the above goals by varying the parameters in the model for DMP-streaming. The reason why we use the model (instead of simulation or empirical study) is that it allows us to systematically explore the parameter space, a task that is difficult when using other means.

The parameters of the TCP flows are set as follows. The loss rate is varied from 0.004 to 0.04. The timeout value is varied from 1 to 4, based on several measurements in [24], and our measurements in Section 6 and [31]. The RTT is in the range of 40 ms to 300 ms based on measurement results that the median RTTs between two sites on the same coast and the two coasts in the US are 50 and 100 ms respectively [14]. 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 k-th path. Throughout this section, we use $K = 2$; performance study under 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. Afterwards, we discuss the benefits from using multiple paths in TCP-based streaming. At the end, we compare DMP-streaming to a static scheme.

7.1 Conditions for satisfactory performance: homogeneous paths

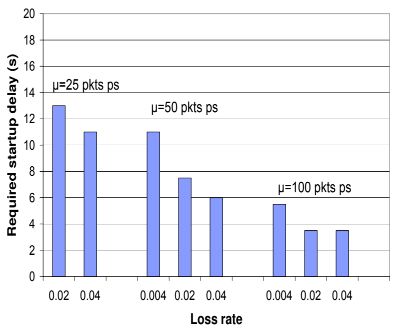
We now consider homogeneous paths and determine the conditions under which DMP-streaming leads to satisfactory performance. A performance is said to be *satisfactory* when the fraction of late packets is less than 10^{-4} for a startup delay of around 10 seconds. This is because people can usually tolerate several seconds of startup delay and studies show that video quality drops when the packet loss rate exceeds 10^{-4} (e.g., [30]).

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.

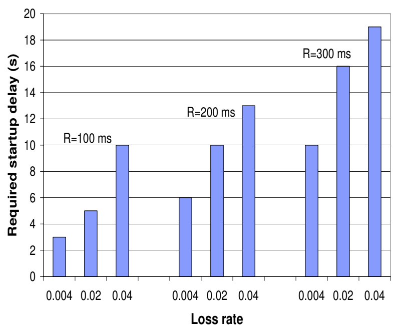
We now vary the value of σ_a/μ from 1.2 to 2.0 to identify the minimum value of σ_a/μ that leads to satisfactory performance. For ease of notation, we drop the index in the subscript and use p, R, T_O and σ to denote respectively the loss rate, RTT, time-out value and achievable TCP throughput on all the paths (since the paths are homogeneous). Let σ_R denote the throughput in one RTT. Then $\sigma_R = \sigma R$ and 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; (2) fixing σ_R (by fixing p and T_O) and R, and varying the playback rate μ. In both manners, 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.

We first present the results when varying σ_a/μ from 1.2 to 2.0 by fixing σ_R and μ, and varying the RTT R. 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 Fig. 8, 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. Fig. 9(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 represent a practical setting). We observe that the required startup delay is around 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 present the results when varying σ_a/μ from 1.2 to 2.0 by fixing σ_R and R, and varying the playback rate μ. The RTT R is set to 100, 200 or 300 ms. Fig. 9(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 generally around 10 seconds except the settings with a large RTT, high loss rate and timeout value (e.g., $p = 0.04$, $T_O = 4$). For those settings, a higher σ_a/μ



(a) μ=25, 50 or 100 pkts per second.



(b) R=100, 200 or 300 ms.

Figure 9: Homogeneous paths: required startup delay so that the fraction of late packets is below 10^{-4}, $T_O = 4$, $\sigma_a/\mu = 1.6$.

(e.g., $\sigma_a/\mu = 1.8$) is required to achieve a satisfactory performance.

7.2 Conditions for satisfactory performance: heterogenous paths

We identify the conditions for satisfactory performance under heterogenous paths by exploring the impact of path heterogeneity — once understanding the impact, we can relate the conditions for homogenous paths to those for heterogenous paths. More specifically, we compare the performances of DMP-streaming under two scenarios. In the first scenario, the 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 all the 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 k-th path. 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, i.e., $\sum_{k=1}^K \sigma_k^e = K\sigma^o$. Although heterogeneous paths may differ in any combination of their parameters, to make the exploration tractable, we focus on two types of heterogeneous paths as follows, where γ represents the extent of path heterogeneity, referred to as *heterogeneity factor*:

- Case 1: the two paths only differ in their RTTs. That is, $p_1^e = p_2^e = p^o$, $T_{O_1}^e = T_{O_2}^e = T_O^o$, $R_1^e = \gamma R^o$, $R_2^e = R^o/(2 - 1/\gamma)$, $\gamma > 1$. 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 [24] to obtain the same aggregate achievable TCP throughput as that under homogeneous paths.

We now report the results for the above two cases. All the settings below 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 RTT. 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. Fig. 10 plots the required startup delay (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 a 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

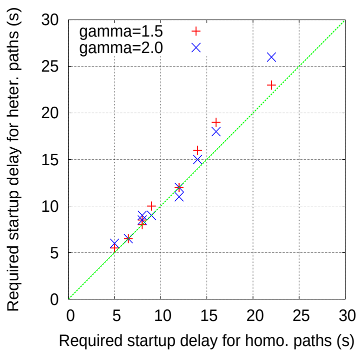


Figure 10: Impact of path heterogeneity.

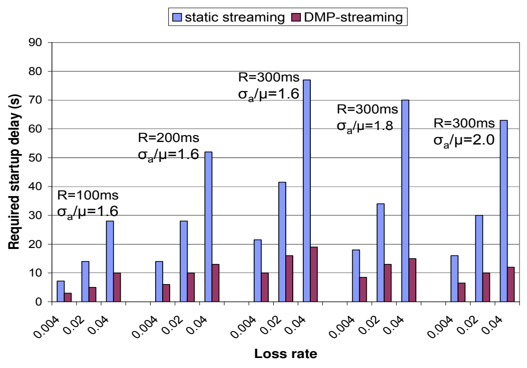


Figure 11: Performance comparison of DMP-streaming and static-streaming, $T_O = 4$.

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 lower than that in single-path TCP-based streaming (which generally requires that the achievable TCP throughput to be twice as the video bitrate [31]). The reasons for this lower requirement is that DMP-streaming dynamically allocates packets onto the multiple paths to take advantage of the 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 alternates between a zero and non-zero throughput, with the period of 10 seconds. The non-zero throughput on path P is 2μ packets per second. The non-zero throughputs on paths P_1 and P_2 are x and $(2\mu - x)$ packets per second, respectively, $x \in (0, \mu]$. Therefore, the average throughput of path P is μ packets per second, equal to the aggregate throughput over paths P_1 and P_2. We show that, for a startup delay of 5 seconds and playback rate of μ packets per second, the average fraction of late packets under DMP-streaming is lower than that under single-path live streaming for all values of $x \in (0, \mu]$ (details in [32]). This is because when paths P_1 and P_2 both start with zero (or non-zero) throughput, the fraction of late packets under DMP-streaming equals to that under single-path live streaming. However, when this is not the case (i.e., the 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 loss.

7.4 Comparison with a static scheme

We now compare DMP-streaming and a static streaming scheme which allocates packets statically onto multiple paths according to the average bandwidths (which are obtained beforehand through measurement) of these paths. Intuitively, DMP-streaming outperforms static-streaming since the allocation under DMP-streaming is according to the *current* network bandwidths. We next demonstrate quantitatively that this is indeed the case.

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. Then we can regard satic-streaming as streaming two separate videos, each with playback rate $\mu/2$, over these two paths. We can therefore obtain the fraction of late packets using our single-path streaming model in [31].

We now compare the results from DMP-streaming and static-streaming in the above scenario. 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. Fig. 11 plots the results from 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.

8. CONCLUSIONS

In this paper, we designed a simple and practical scheme, DMP-streaming, 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.

9. REFERENCES

[1] *Planetlab*. http://www.planet-lab.org/.

[2] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of TCP/IP with stationary random losses. In *SIGCOMM*, pages 231–242, 2000.

[3] J. Apostolopoulos, T. Wong, W. Tan, and S. Wee. On multiple description streaming with content delivery networks. In *IEEE INFOCOM*, 2002.

[4] S. Bohacek. A stochastic model of TCP and fair video transmission. In *Proc. IEEE INFOCOM*, 2003.

[5] N. Cardwell, S. Savage, and T. Anderson. Modeling TCP latency. In *INFOCOM (3)*, pages 1742–1751, 2000.

[6] J. Chesterfield, R. Chakravorty, I. Pratt, S. Banerjee, and P. Rodriguez. Exploiting diversity to enhance multimedia streaming over cellular links. In *Proc. IEEE INFOCOM*, Miami, FL, March 2005.

[7] P. de Cuetos, P. Guillotel, K. W. Ross, and D. Thoreau. Implementation of adaptive streaming of stored MPEG-4 FGS video over TCP. In *International Conference on Multimedia and Expo*, August 2002.

[8] P. de Cuetos and K. W. Ross. Adaptive rate control for streaming stored fine-grained scalable video. In *Proc. of NOSSDAV*, May 2002.

[9] E. de Souza e Silva and R. M. M. Leao. The TANGRAM-II environment. In *Proc. of the 11th Int. Conf. on modeling tools and techniques for computer and communication system performance evaluation*, May 2000.

[10] D. R. Figueiredo, B. Liu, V. Misra, and D. Towsley. On the autocorrelation structure of TCP traffic. *Computer Networks Journal*, 2002.

[11] L. Golubchik, J. Lui, T. Tung, A. Chow, W. Lee, G. Franceschinis, and C. Anglano. Multi-path continuous media streaming: What are the benefits? *Performance Evaluation*, 2002.

[12] F. Guo, J. Chen, W. Li, and T. cker Chiueh. Experiences in building a multihoming load balancing system. In *Proc. IEEE INFOCOM*, 2004.

[13] H.-Y. Hsieh and R. Sivakumar. A transport layer approach for achieving aggregate bandwidths on multi-homed mobile hosts. In *MOBICOM*, Atlanta, GA, September 2002.

[14] B. Huffaker, M. Fomenkov, D. Moore, and K. Claffy. Macroscopic analyses of the infrastructure: Measurement and visualization of Internet connectivity and performance. In *A Workshop on passive and active measurements*, April 2001.

[15] D. Jurca and P. Frossard. Packet selection and scheduling for multipath video streaming. *IEEE Transactions on Multimedia*, February 2006.

[16] T. Kim and M. Ammar. Receiver buffer requirements for video streaming over TCP. In *Proc. of Visual Communications and Image Processing Conference*, January 2006.

[17] C. Krasic and J. Walpole. Priority-progress streaming for quality-adaptive multimedia. In *ACM Multimedia Doctoral Symposium*, Ottawa, Canada, October 2001.

[18] M. Li, M. Claypool, R. Kinicki, and J. Nichols. Characteristics of streaming media stored on the Internet. Technical Report WPI-CS-TR-03-18, CS Department, Worcester Polytechnic Institute, May 2003.

[19] Y. J. Liang, E. G. Steinbach, and B. Girod. Real-time voice communication over the Internet using packet path diversity. In *ACM Multimedia*, Ottawa, Canada, September/October 2001.

[20] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the TCP congestion avoidance algorithm. *Computer Communications Review*, 27(3), 1997.

[21] M. Mellia, I. Stoica, and H. Zhang. TCP model for short lived flows. *IEEE Communication Letters*, 6(2), February 2002.

[22] T. P. Nguyen and Z. Avideh. Mutiple sender distributed video streaming. *IEEE Transaction on Multimedia*, 6(2):315–326, April 2004.

[23] J. Padhye, V. Firoiu, and D. Towsley. A stochastic model of TCP Reno congestion avoidance and control. Technical Report 99-02, Department of Computer Science, University of Massachusetts, Amherst, 1999.

[24] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP throughput: A simple model and its empirical validation. In *Proc. ACM SIGCOMM*, pages 303–314, Vancouver, CA, 1998.

[25] R. Rejaie and A. Ortega. PALS: Peer-to-peer adaptive layered streaming. In *NOSSDAV*, 2003.

[26] B. Ribeiro, E. de Souza e Silva, and D. Towsley. On the efficiency of path diversity for continuous media applications. Technical Report 05-19, Department of Computer Science, University of Massachusetts, Amherst, 2005.

[27] N. Seelam, P. Sethi, and W. chi Feng. A hysteresis based approach for quality, frame rate, and buffer management for video streaming using TCP. In *Proc. of the Management of Multimedia Networks and Services*, 2001.

[28] K. Sripanidkulchai, B. Maggs, and H. Zhang. An analysis of live streaming workloads on the Internet. In *IMC*, pages 41–54, 2004.

[29] J. van der Merwe, S. Sen, and C. Kalmanek. Streaming video traffic: Characterization and network impact. In *Proc. of International Web Content Caching and Distribution Workshop*, August 2002.

[30] O. Verscheure, P. Frossard, and M. Hamdi. MPEG-2 video services over packet networks: Joint effect of encoding rate and data loss on user-oriented QoS. In *Proc. of NOSSDAV*, July 1998.

[31] B. Wang, J. Kurose, P. Shenoy, and D. Towsley. Multimedia streaming via TCP: An analytic performance study. In *Proc. ACM Multimedia*, October 2004.

[32] B. Wang, W. Wei, Z. Guo, and D. Towsley. Multipath live streaming via TCP: Performance and benefits. Technical Report BECAT/CSE-TR-06-7, Computer Science and Engineering Department, University of Connecticut, 2006.

[33] Y. Wang, M. Claypool, and Z. Zuo. An empirical study of video performance across the Internet. In *ACM SIGCOMM Internet Measurement Workshop*, San Francisco, CA, November 2001.