# Global 1-Mbps Peer-Assisted Streaming: Fine-Grain Measurement of a Configurable Platform

Joe Wenjie Jiang, S.-H. Gary Chan, *Senior Member, IEEE*, Mung Chiang, *Fellow, IEEE*, Jennifer Rexford, D. Tony Ren, and Bin Wei

***Abstract*—High-resolution video is defining a new age of peer-assisted video streaming over the public Internet. Streaming over 1-Mbps videos in a scalable and global manner presents a challenging milestone. In this work, we examine the feasibility of 1-Mbps streaming through a global measurement study. In contrast to previous measurement studies that crawl commercial applications, we conduct fine-grain, controlled experiments on a configurable platform. We developed and deployed FastMesh-SIM, a novel peer-assisted streaming system that leverages proxies, scalable streaming trees and IP multicast to achieve 1-Mbps streaming at a global scale.**

**With the configurability-enabled design, we are allowed to conduct controlled experiments by varying design decisions under a wide range of operating conditions, and measuring in-depth, fine-grain metrics at a per-hop, per-segment level. We collected hundreds of hours of streaming traces that broadcast live TV channels to more than 120 peers and 30 proxies, with a global geographic footprint over 8 different countries. Data analysis demonstrates how a set of design decisions collectively overcome the 1-Mbps barrier. The various operational issues we uncovered provide insights to service providers that want to deploy a commercial system at a larger scale and a higher streaming rate. By comparing theory and practice, we also confirm theory-inspired architectural decisions, and show that our system indeed achieves throughputs close to theoretical upper-bound calculated under many ideal assumptions.**

***Index Terms*—Communications technology, communication systems, computer networks, peer-to-peer computing.**

## I. Introduction

MOST existing commercial peer-assisted streaming systems [1] deliver a streaming rate at a few hundreds of kbps, which does not meet the growing demand for high-resolution videos. A decent viewing experience on today's home TV or iPad screen usually requires a sustained streaming rate on the order of Mbps. While 1-Mbps streaming is happening today, it often relies on the pervasive deployment of expensive infrastructure such as content distribution networks (CDNs), or is offered to a limited number of premium users only. Providing high-quality, reliable service to a large user population at a reasonable cost is the lifeblood to today's service operators.

It is commonly believed that the conventional P2P approach does not sustain a high streaming rate due to insufficient bandwidth resources. A number of architectural choices were proposed to improve the system performance and scalability, including the use of proxies or helper nodes [2], packing multiple trees [3], IP-multicast integration [4], etc. Other design decisions, ranging from the segment size, and the scheduling policy, to parallel TCP connections, also have a great impact on the performance and reliability. In order to examine whether these ideas collectively push the envelope to above 1 Mbps under realistic Internet conditions, and which decisions are the key factors that affect the performance, we need a *systematic* and *quantitative* understanding of these design choices. This study is driven by the following requirements:

- a highly instrumented and configurable platform;
- fine-grain and detailed measurement data;
- reproducible results that do not rely on proprietary commercial applications;
- underlying network support such as IP-multicast capability;
- dedicated resources in each peer that isolate performance disruption.

This paper is about the deployment, experiment, measurement, and data analysis over an operational system satisfying all of the above needs. The deployed *FastMesh-SIM* is a novel *push*-based P2P streaming system that consists of multiple trees—as suggested by recent theoretical studies on P2P streaming capacity [3], [5], [6]—to offer high-quality streaming at a global scale, in contrast to other studies that primarily focus on pull-based systems. The two recurring themes of this paper are: *1) engineer 1-Mbps peer-assisted streaming and 2) learn from fine-grain measurement through controlled experiments.* Through statistical studies of the data, we also have the opportunity to uncover many practical issues in operating a global streaming service.

### A. Enabling Fine-Grain Measurement of a Configurable Platform

Identifying the obstacles to 1-Mbps streaming requires new methodologies to obtain a finer-grained understanding of the system bottlenecks. New metrics need to be introduced to measure and identify the effectiveness for every design decision we make. In addition, a distributed system providing a global-scale,

Manuscript received August 15, 2011; revised January 13, 2012; accepted April 04, 2012. Date of publication April 25, 2012; date of current version September 12, 2012. This work was supported in part by AFOSR MURI grant FA9550-09-1-0643 and in part by the General Research Fund from the Research Grant Council of the Hong Kong Special Administrative Region, China (611209). The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Christophe De Vleeschouwer.

J. W. Jiang and J. Rexford are with the Department of Computer Science, Princeton University, Princeton, NJ 08540 USA (e-mail: wenjiej@princeton.edu; jrex@princeton.edu).

S.-H. G. Chan and D. T. Ren are with the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong (e-mail: gchan@cse.ust.hk; tonyren@cse.ust.hk).

M. Chiang is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08540 USA (e-mail: chiangm@princeton.edu).

B. Wei is with AT&T Labs Research, Florham Park, NJ 07932 USA (e-mail: bw@research.att.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2012.2196509

resource-demanding service is more sensitive to the behavior of underlying communication network and customers, and noise can unintentionally amplify or diminish the importance of certain design parameters. Measurement in a controlled environment allows us to focus on one set of factors in each experiment while controlling for other elements that may affect the system performance.

Previous measurement studies usually consist of "black box" characterizations of proprietary, commercial P2P streaming services [7]–[12]. Complementary to prior work, we focus on the following approach: controlled experiments on a configurable P2P streaming system that is designed, implemented, and deployed by the same team that carried out the measurement study. This enables us to collect fine-grained measurements, e.g., a rich set of per-segment per-hop timestamp information, under a wide range of design choices, while running experiments over the public Internet. Our goal, then, is to gain qualitative and quantitative understandings of how major design decisions and environmental conditions affect the streaming performance, i.e., rate, delay, and reliability.

### B. Experience With Realizing Global 1-Mbps Streaming

The in-depth measurement study is conducted on our hybrid proxy-P2P streaming system, FastMesh-SIM [13], [14], which broadcasts live TV channels at a 1-Mbps quality, with a large geographic footprint over 8 countries in 5 continents. We analyzed more than 200 h of streaming logs from over 20 experimental trials, including more than 120 peers and 30 proxies. The rich dataset collected from our tightly controlled experiments in a highly instrumented system offers a unique perspective into many design choices proposed *separately*. In particular, we summary our main insights as follows:

*1) Capability of Proxy Nodes:* We revisit the common wisdom that the pure P2P approach does not sustain a high streaming rate due to insufficient peer bandwidth. Many studies proposed the use of proxies or hybrid CDN-P2P for cost-effective streaming. We carefully quantify the capacity of proxy nodes needed to meet the 1-Mbps requirement, and demonstrate that a small population of "super" nodes (e.g., campus LAN users) is competent for the proxy functionality, in contrast to other work that suggest dedicated servers with a high bandwidth.

*2) Scalability of a Two-Tier Architecture:* Scalability is the key requirement for any service provider who wishes to provide consistent and reliable high-rate streaming. The proposed two-tier architecture that separates the design space into groups of peers clustered by geographic proximity—a carefully optimized proxy mesh (FastMesh) at the core, and simple resilient trees (SIM) at the peripheral with layer-3 multicast support—is a promising approach to attain a high streaming rate among a reasonably large population of users. The 120 peers in our experiment, are supported by only 5 proxies located at two campuses, with each proxy serving tens of peers. A straightforward back-of-the-envelop calculation extends the system scale to close to a thousand peers with our current proxy network size.

*3) Complexity of Tree Construction:* A number of recent literatures [3], [5], [6], [15] studied how to achieve the P2P streaming capacity, i.e., the maximum streaming rate that can be supported given a set of peers. Most solutions propose sophisticated tree packing algorithms that require a large number of trees. In this work we show that a simple mesh construction that involves a limited number of trees work well in practice, and as demonstrated in our experiment achieves over 85% of the streaming rate upper-bound.

*4) Optimizability of Streaming Delay:* The user-perceived streaming delay and jitters are also important performance metrics that service providers care about. We characterize various delay components, and show that the wide-area (propagation) delay is dominated by other components such as transmission and protocol delays that are affected by many design choices like the scheduling policy and segment size. We quantify how to carefully tune these parameters to improve the delay bound.

*5) Reliability Under Varying Network Conditions:* Through a highly instrumented traffic monitor, we are able to identify and traceback the performance degradation that happens at a particular hop and segment. This gives us a unique opportunity to improve the system reliability by dissecting the performance bottleneck, e.g., edge versus core of the network, which would be difficult without a fine-grain measurement.

Our work has the obvious limitation that some of the quantitative results may depend on specific design and implementation choices in our system, or the details of our current deployment. However, experimenting with an operational, integrated system with ample geographic diversity provides valuable experiences to operators who wish to deploy similar services and revisit many design choices that were proposed previously as a joint solution. The rest of the paper is organized as follows. Section II gives an overview of the design and implementation of the FashMesh-SIM system. Section III presents the fine-grained measurement methodology and the design parameters we vary in the experiments. Sections IV–VI, the core sections, present the analysis of the measurement data and our answers to the questions raised above. Section VII discusses the theory-practice similarities and discrepancies illuminated by this study, and the lessons we learned through the interaction between the development engineers and measurement researchers. We further discuss related work in Section VIII, and conclude in Section IX.

## II. Design of FastMesh-SIM

We briefly review our design choices in FastMesh-SIM aimed at improving the performance metrics of streaming rate (i.e., throughput), delay, and scalability. FastMesh-SIM constructs application-level multicast trees. As illustrated in Fig. 1(a), it consists of two push-based protocols: 1) FastMesh that constructs a streaming backbone among an upper tier of more stable proxies with low delay, and 2) SIM (scalable island multicast) by which a lower tier of end-user peers self-organize into one or multiple trees, leveraging local IP multicast support wherever possible. While FastMesh and SIM have been separately reported in prior publications [13], [14], this paper reports the first deployment of an integrated streaming service of FastMesh-SIM and the first in-depth global measurement

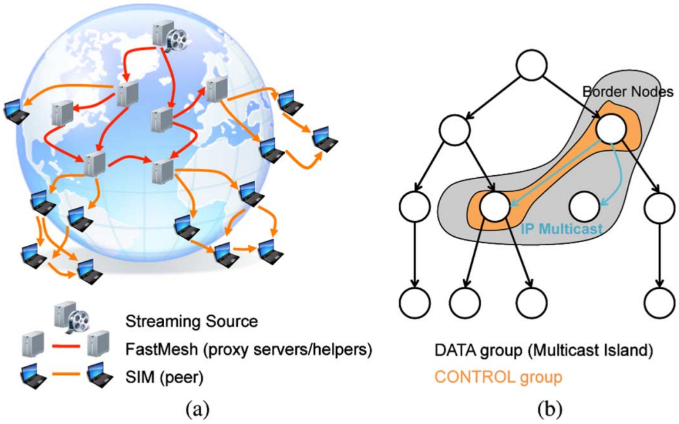


Fig. 1. System designs of FashMesh-SIM. (a) Proxy-based peer-assisted two-tier architecture. (b) Combining IP-multicast with overlay tree streaming in SIM.

study of the system. The system design consists of the following key concepts:

*1) Push-Based Stream Delivery:* FastMesh-SIM is a pure push-based system that constructs one or more application-level multicast trees. The video is divided into multiple sub-streams, each of which is delivered by a tree. The tree structure is maintained at every peer in a distributed manner. As segments in a substream arrive, parent nodes immediately push the data to their children.

*2) Low-Delay Proxy Mesh:* Each video substream is sent through the proxies before reaching the lower tier of peers. The proxies are stable and bandwidth-rich machines, such as dedicated infrastructure nodes contributed by the content provider. The proxies run the fully distributed protocol FastMesh [14], which builds multiple high-bandwidth low-delay spanning trees. The protocol is adaptive such that the delay is constantly optimized given any change of server availability. As the number of proxies may change over time, FastMesh continues to perform local network measurements and adaptations in order to construct trees that minimize the worst-case delay in delivering data from the servers, subject to constraints on upload capacity.

*3) Scalable Local Streaming Trees:* A swarm of peers close to a proxy server run the lightweight SIM [13] protocol to construct low-delay trees with minimal overhead. As a new peer joins the system, it contacts a rendezvous point (RP) that returns its local proxy, which in turn gives a random list of existing peers rooted at the proxy to bootstrap the process. The peer selects a set of nodes as its parents, taking both RTT and bandwidth into consideration. The tree formation is distributed and adaptive, as the underlying network conditions and node availability change over time.

*4) IP-Multicast Integration:* A unique feature of SIM is to integrate IP-multicast with application-layer multicast to improve bandwidth efficiency. Although IP multicast is not globally available on today's Internet, many local area networks are multicast-capable. SIM capitalizes on local support for IP multicast by embedding these "multicast islands" into the overlay streaming trees, as illustrated in Fig. 1(b). Peers within a multicast island receive data using IP multicast, and communicate with the outsiders through border nodes on streaming trees. SIM has a distributed mechanism to decide which multicast nodes are in an island so as to eliminate duplications, thus improving the efficiency of bandwidth usage in a local network.

| | | | | |
|---|---|---|---|---|
| `peer_id` | `seg_id` | `gen_t` | `rcvd_t` | `send_t` |

Fig. 2. Format of segment delay timestamps.

*5) Reactive Error Recovery:* FastMesh is able to adapt to server churns by re-optimizing the mesh. SIM implements a more sophisticated error-recovery mechanism to respond to temporary or unexpected packet or stream losses [16]. Through backup parents and distributed tree re-formation, SIM achieves fast error recovery, even under frequent peer churns.

## III. Measurement Metrics and Methodology

Conducting experiments with the FastMesh-SIM system enables both visibility (through fine-grained instrumentation of the software) and control (by configuring many tunable design parameters).

### A. Visibility: Fine-Grained Metrics Instrumentation

Traditional measurement studies in P2P streaming often employ a large-scale user pool generating real traffic on the Internet. However, due to the large user population and peer churns, they often rely on random or statistical sampling, which provides a view that is partial or coarse-grain. As an alternative approach, we use a relatively smaller set of peers over diverse geographic locations, which allows us to perform an in-depth examination of *every* peer and *every* segment throughout all experiments, while not losing the geographic properties of large systems. We install a built-in monitor for each peer that is able to observe packet-level activities.

Our monitor generates a record for every segment it receives. A segment record contains several timestamps about the segment's lifetime, as shown in Fig. 2. The fields of `peer_id` and `seg_id` specify which peer and which segment this record belongs to. `gen_t` specifies when the segment is generated at the source. `rcvd_t` records when the segment is received at the local peer. `send_t` records when the segment is scheduled to be sent (and to which peer). A peer's local clock is synchronized by the NTP servers to ensure the validity of the timestamp. Together with the data payload, the record is sent to the next hop peer and used to infer more delay information. By collecting the records from all peers, we can reconstruct the lifetime for every segment in the network.

Besides the segment delay log, our monitor also records a peer's local information periodically, e.g., every 10 s. Fig. 3 lists the most important quantities we recorded. The measured and inferred quantities, in general, can be grouped into the following five categories:

*1) Node:* The peer's ID, IP address, port number, and the time that the record is generated.

*2) Topology:* The peer's parent list `PL`, child list `CL`, the longest path from the source `PATH` and its depth `DP`. `DEG` records the incoming degree and outgoing degree. All the information

| Metrics | Measured quantities | Inferred quantities |
|---|---|---|
| **Node** | ID, IP, PORT, TIME | – |
| **Topology** | PL, CL | DP, PATH, DEG |
| **Bandwidth** | UD, DD (U/M/R), UT, DT | UB, DB |
| **Buffer** | FS, LS, PH | CI |
| **Delay** | PB, TX, PR | SC |

Fig. 3. Fine-grained measurement quantities.

helps track how the overlay topology changes during churns and failovers.

*3) Bandwidth:* The peer's upload/download data bytes UD/DD transmitted during a monitoring interval, e.g., 10 s. We can further differentiate various data sources, i.e., overlay unicast DDU, IP-multicast DDM or recovery server DDR. UT/DT is the total upload/download traffic, including data payload and control messages. UB/DB are the average peer upload and download bandwidth derived by dividing the data bytes by the length of a monitoring interval.

*4) Buffer:* A snapshot of the current playback buffer. FS is the first (earliest) segment. LS is the last (latest) segment. PH is the current segment fed into the player. CI is the continuity index, defined as one minus the segment loss rate, i.e., the number of holes divided by the buffer length.

*5) Delay:* With the segment delay records, we can measure or infer a set of delay components: playback delay PB, transmission delay TX, propagation delay PR and scheduling delay SC. The precise definition of these delay quantities are given below. All collected quantities are averaged over the one monitoring period, e.g., 10 s by default.

*Definition 1:* Playback (or end-to-end) delay PB is the time elapsed from a segment's generation at the source, to its complete receipt (including error recovery) at the local peer, i.e.,

$$\texttt{PB} = \texttt{rcvd\_t} - \texttt{gen\_t}.$$

*Definition 2:* Transmission delay TX is the total elapsed time from when a segment is scheduled to be sent, to the completion of this transmission, over all hops the segment traverses, i.e.,

$$\texttt{TX} = \sum_{hop\ i} \texttt{rcvd\_t}_{\mathrm{child}(i)} - \texttt{send\_t}_{\mathrm{parent}(i)} - RTT_i.$$

*Definition 3:* Scheduling (or protocol) delay SC is the total waiting time before a received segment is scheduled to be sent, over all hops the segment traverses, i.e.,

$$\texttt{SC} = \sum_{hop\ i} \texttt{send\_t}_{\mathrm{parent}(i)} - \texttt{rcvd\_t}_{\mathrm{parent}(i)}.$$

We can immediately validate the following relationship for the above defined playback, propagation, transmission and scheduling delay, and this is how we infer SC from (PB, PR, TX) measurements. Since $\texttt{rcvd\_t}_{\mathrm{source}} = \texttt{gen\_t}$, we have

$$\texttt{PB} = \texttt{PR} + \texttt{TX} + \texttt{SC}.$$

With this information above, we can reconstruct the lifespan of every segment.

### B. Control: Configurable Streaming Platform

In this measurement study, we study design space by intentionally turning on/off some features and tuning system parameters. We present a taxonomy of the "control knobs":

*1) Experimenting With Architectural Choices:*

- *Layer-3 support:* IP-multicast can be enabled or disabled. We let a fraction of peers to be IP-multicast capable, either on the same local network, or across different networks.
- *Proxy functionality:* A proxy node can play two roles: proxy server and proxy helper, and their difference will be explained later. This knob controls the achievable streaming rate given a certain number of proxies.

*2) Varying Degrees of Design Freedom:*

- *Parallel connections:* We allow a peer to set up multiple TCP connections for data transmission. Enabling multiple TCP connections may overcome the TCP throughput limitation over one single long-haul connection.
- *Segment size:* Segment size is the smallest replication unit used in a P2P system. It determines how quickly a peer can disseminate the data it receives. The segment size knob controls the tradeoff between the transmission delay and the protocol overhead.

*3) Synthesizing Streaming Environment:*

- *Peer churns:* We synthesize churns by letting peer behaviors follow our pre-programmed script. We consider two types of dynamics, peer churns and server churns. Tuning the knob to allow different churn rates affects the system dynamics and reliability.
- *Streaming workload:* We tune the video rate such that the system operates under different workloads, over the wide range of 100 kbps to 4 Mbps. This allows us to explore the limit of streaming capacity over the Internet.
- *Node access:* Nodes, which can be proxies or peers, are selected from different ISPs or locations (e.g., continents). Peers are campus users or residential home users. Nodes can be dedicated machines with excellent network access, or virtual machines from resource-shared platforms.

These knobs together allow us to perform a set of controlled experiments and study the design tradeoffs, by decoupling a large number of factors.

### C. Deployment

In our trials, we deploy 34 standard commodity laptops and desktop machines with dedicated bandwidth as proxies, located on 7 collaborator sites on different continents. In one trial, we also select a few PlanetLab nodes to allow us to compare the performance of two types of nodes: those with dedicated and those with shared resources. All of them serve as proxy servers (or helpers) in our experiments, and run the FastMesh protocol. We also employ 100 desktop machines on the campus network from two collaborators: Hong Kong (China) and Princeton (US). These machines serve as peers and run the SIM protocol. They are connected to the Internet via campus LAN, and with IP-multicast enabled. We also invite 20 residential broadband/DSL home users from Hong Kong to participate in these experiments.

The geographic locations of the proxies and peers are shown in Fig. 4, and node statistics are given in Table I. Although we do

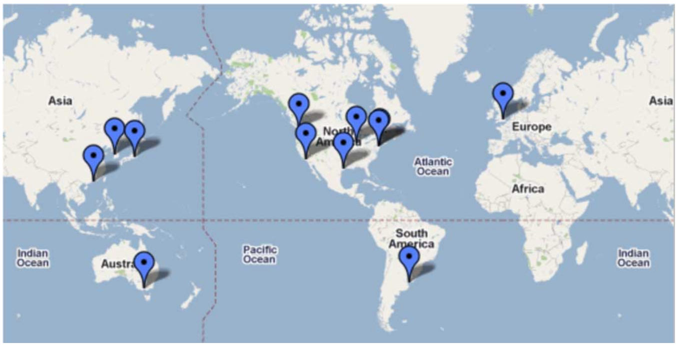


Fig. 4. Global deployment of proxy nodes.

TABLE I
GEOGRAPHIC LOCATIONS AND NODE ACCESS PROPERTIES OF PROXY SERVERS AND PEERS USED IN OUR EXPERIMENTS

| Nodes | **Proxies** | | **Peers** | |
|---|---|---|---|---|
| Total # | 42 | | ∼120 | |
| Type | Dedicated | PlanetLab | Dedicated | Residential |
| Location | Princeton 12<br>HK 7<br>Caltech 3<br>Uruguay 3<br>Korea 3<br>Australia 3<br>UK 3 | US 6<br>Japan 2 | Princeton<br>30<br>HKUST<br>70 | HK<br>20 |
| Bandwidth | 10Mbps | shared | 10Mbps | varied |
| IP-multicast | na | na | enabled | na |

not employ a large number of peers in our experiments, having around 50 peers in a local swarm, e.g., close to the same proxy server, represents a reasonable size of a practical scenario. A higher-level mesh network with tens of proxies also provides a good starting point to study proxy deployments and steaming performance at a global scale.

The data analyzed in this work are collected in more than 20 experiment trials conducted during December 2009–September 2010 that broadcast live TV channels (with the streaming source in Hong Kong), which in total contribute a set of traces of more than 200 h.

## IV. ACHIEVING 1-MBPS STREAMING RATE

In the following three sections, we analyze the data collected from the experiment trials and draw lessons on building a global 1-Mbps P2P streaming platform. We first investigate how to achieve 1-Mbps streaming rate by collectively applying a set of techniques. For entertainment-grade user experience, delay and reliability are important metrics. We further investigate the user playback delay and discuss how to minimize delay by differentiating various delay components in the next two sections. We also study the system reliability under churning behaviors and varying network conditions.

### A. Capability as Proxy Nodes

We first show the capability of the FastMesh protocol to achieve 1-Mbps streaming rate among a set of geographically diverse proxy nodes. We select 15 nodes to function as proxies, 10 of which are dedicated servers from our collaborators and 5 from PlanetLab. We also employ PlanetLab nodes in order to contrast resource-dedicated and resource-shared environments. The geographic coverage is shown in Fig. 4, which represents a typical distribution of proxies with diverse RTTs in the wide area.

Fig. 5 shows the correlation between a node's streaming capability and its geographic location. Fig. 5(a)–(c) shows the mean and variance of the received data rate, uploaded data rate and RTT distribution of proxy nodes during a 1-h trial, with nodes grouped by their ISPs and indexed in an increasing order of received data rates.[1] Seven out of 15 nodes achieve a persistent 1-Mbps streaming rate. Other nodes suffer from rate fluctuations and some even receive less than 80% of the required bit-rate. We also measure the RTTs between a node and its most distant parent, as well as the source (placed in Hong Kong). Comparing these figures leads to the observation that the streaming rate has a strong correlation with a node's physical location, e.g., the streaming rate degrades sharply as RTTs exceed 100 ms. As the throughput of long-haul TCP connections are greatly affected by RTTs, the wide-area distance turns out as an important factor in achieving a high streaming rate. In particular, the cross-continent connections are severely impaired. As we will show later, the use of parallel connections and helper nodes is able to rectify such a problem.

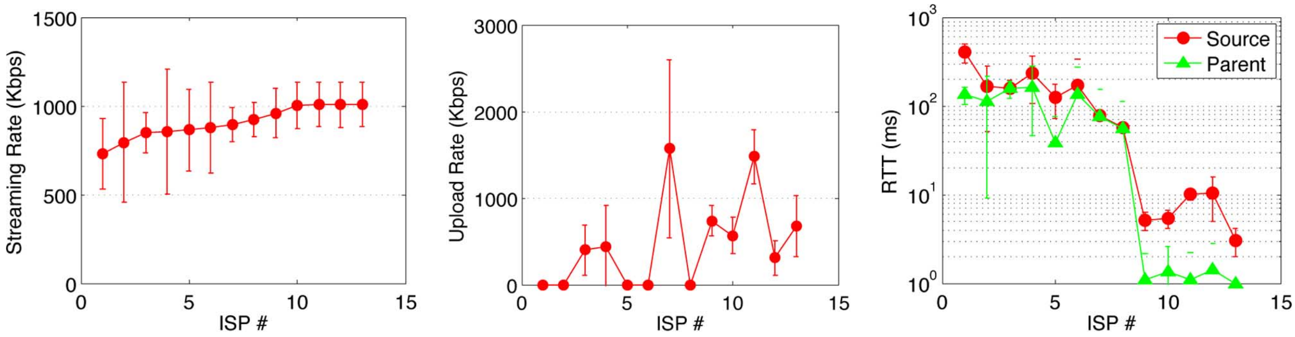
It is interesting to note that nodes with the poorest performance are from PlanetLab—all of them achieve less than 800 kbps rate on average. Bandwidth cap and sharing CPU cycles with other applications understandably put the sustained 1-Mbps video streaming at a risk. The FastMesh protocol should avoid using these unstable proxies to deliver video streams to others. Fig. 5(b) shows that most uploading workload is assigned to "good" proxies, so the bandwidth from resource scarce nodes can be reserved to support their local peers.

### B. Reliable Tree-Based Streaming

We next study the influence of residential peers with slower and less reliable Internet access in the global 1-Mbps streaming experiment. A swarm of peers in each location get the complete set of substreams from one or two nearby proxy servers, and forward the stream within themselves by constructing an application layer multicast tree. In this trial, we deploy three proxy servers in Hong Kong and let a mixture of 30 campus users and 20 residential broadband/DSL users join the streaming channel. We perform two sets of experiments, one with 500-kbps streaming rate and the other with 1-Mbps streaming rate, each lasting 45 min. The results are presented in Fig. 6.

In the 500-kbps experiment, both campus and residential users achieve a very steady streaming rate, which demonstrates the robustness of the constructed streaming tree. However, they make significantly distinct contributions to the system in terms of their uploading capability. Most campus users serve as internal nodes on the tree due to their stable connectivity on both uplink and downlink. Residential DSL users usually have poor uplink capacity and become leaf nodes as the system stabilizes. A few residential users with sufficient resources can achieve up to 2-Mbps uplink capacity and therefore play an important role in an environment when few campus users are present.

[1] Two of the 13 ISPs contain two proxy nodes, and hence there are in total 15 nodes.



(a) (b) (c)

Fig. 5. FastMesh achieves 1-Mbps streaming among a mixture of dedicated servers and PlanetLab nodes at a global scale. Proxy servers are categorized by their locations, ordered with increasing received streaming rate: (a) streaming rate, (b) data uploading rate, and (c) proxy RTT distribution. Resource dedication and RTT bias are important factors that affect the streaming performance.

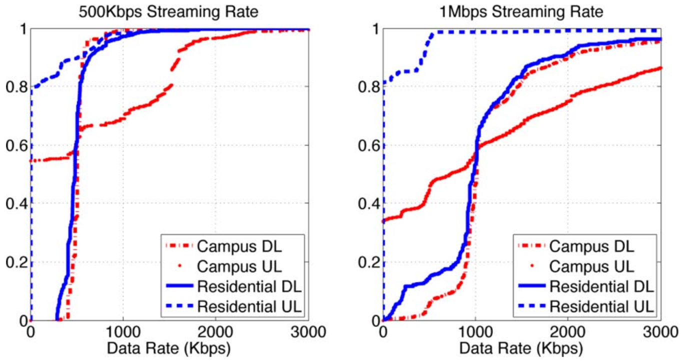
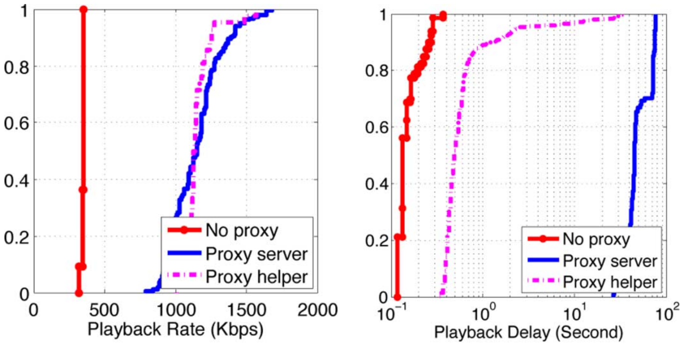


Fig. 6. Achieving 1-Mbps streaming by constructing a SIM tree among a mixture of campus and residential users. Campus and residential users show distinct distributions of upload and download data rates.



(a) (b)

Fig. 7. Using proxies significantly increases the streaming capacity: (a) received stream rate and (b) playback delay. With proxy helpers, the streaming rate is more stable and large playback delays are greatly mitigated.

As we increase the streaming rate to 1 Mbps, the performance of campus users is less impaired, because campus users are able to absorb the workload gracefully among themselves. The performance degradation is expected since there are only three proxy servers which are desktops. This problem would be alleviated as more servers are deployed, though we have already seen encouraging cost-savings by having 3 standard desktop servers support over 50 peers.

### C. *Offloading Workload With Proxy Helpers*

The FastMesh protocol allows a proxy to play two roles: proxy server and proxy helper. A proxy server has to receive the *complete* set of video substreams so it appears as a local server to peers that run the SIM protocol. However, the diverse locations at a global scale and the temporary bandwidth shortage often make it difficult to support a demanding workload by themselves, as demonstrated in Fig. 5. A proxy *helper*, which only requires a partial set of all substreams, provides path diversity to serve the proxy servers by forwarding video substreams, while only consuming a limited amount of resources. This improves the streaming rate through bandwidth aggregation. Peers only receive video streams from their local proxy servers, but do not talk to proxy helpers directly.

We compare the use of proxy servers and proxy helpers, and results are shown in Fig. 7. We deploy proxies with a location distribution as shown in Fig. 4. We conduct two sets of experiments: the first one involves proxy servers only, and the second one involves a 50–50 mixture of proxy servers and proxy helpers. We also compare with the approach without using any proxies, e.g., via the direct SIM tree. All experiments last 30 min. Apparently, the achievable streaming rate without using proxies is up to 300 kbps only, while using proxies greatly improve the streaming rate to 1 Mbps. However, employing a higher streaming rate may suffer from larger playback delays because the throughput becomes more sensitive to various network conditions, as we will show in details later. On the other hand, proxy helpers outperform proxy servers, as the achieved streaming rate is more stable and the playback delay is of several orders of magnitude lower. While we demonstrate that employing both proxy servers and proxy helpers presents a promising solution, calculating an optimal mixture between the two in practice remains an open research problem [5].

### D. *Improving Scalability With IP Multicast*

The measurement data demonstrates that IP-multicast can be seamlessly integrated into overlay streaming trees, even under peer churns. We deploy churning peers in a campus network where IP-multicast support is enabled by network administrators. We intentionally configure some peers to be IP-multicast capable, while turning off others. Because IP-multicast relies on UDP which does not offer retransmission mechanism, we

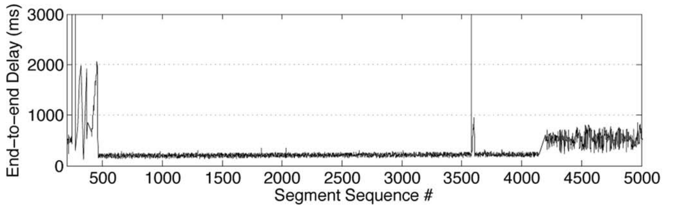


Fig. 8. Embedding IP-multicast into overlay trees: integrating two data planes when one or another starves.

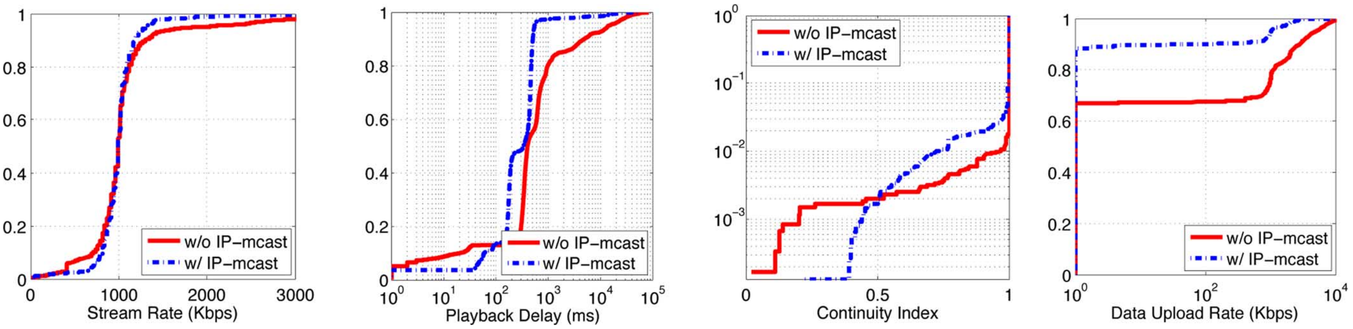


Fig. 9. IP-multicast improves system reliability, scalability, and efficiency.

need to recover the lost IP-multicast data from other sources. The SIM protocol allows a peer to recover data from its overlay parent once the IP-multicasted data is lost. As such, a peer's performance does not rely on one single data plane, and is more robust against highly varying network conditions. In Fig. 8, we show segment-wise delays, and the data source in an experiment with 70 peers from campus nodes. Clearly, the overlay data can immediately bridge the gap due to IP-multicast packet loss. The impact of IP-multicast loss is mitigated so the segment delay is rarely affected. In fact, we are able to keep the system overhead very low, as the amount of overlay data is insignificant compared to the IP-multicast data.

We next examine the benefit of IP-multicast in terms of streaming rate, segment playback lag, buffer continuity index, and uplink usage. We conduct two experiments, one with IP-multicast capable peers, and the other without. We deploy the same set of nodes and allow peer churns with the same parameters. We compare their performances in Fig. 9.

While both approaches achieve an average streaming rate around 1 Mbps, IP-multicast helps to reduce the rate fluctuation, confirming the intuition that IP-multicast is less sensitive to peer churns. Delay reduction by IP-multicast is very significant. Compared to the overlay multicast, delays of IP-multicast are greatly reduced in two ways: lower protocol overhead (e.g., tree reformation, system overhead on segment store-and-forward) and lower bandwidth requirements from the more efficient multicast tree topology. The second factor is especially important, when a higher streaming rate is required.

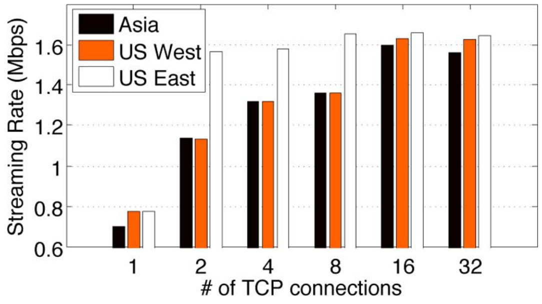


Fig. 10. Experimenting Mbps streaming with multiple TCP connections: finding the optimal number of parallel connections.

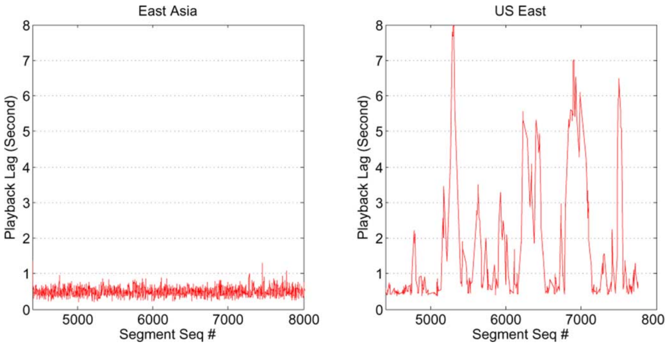


Fig. 11. Tracing segment-level delays and root cause analysis of delay jitters.

IP-multicast also improves the buffer continuity index, except for a very small number of cases in which the buffer continuity is low, e.g., below 0.5, due to the unreliable UDP transmission. The saving of peer uplink bandwidth by IP-multicast is also significant. With IP-multicast, 90% of the time, a majority of peers do not need to upload any data, while in the other case, more than 30% of the time, i.e., without IP-multicast support, some peers contribute more than 1 Mbps of their uplink bandwidth. The amount of bandwidth saving is attained when there are peer churns. Other experiments show that such benefit is even more significant when peers are stable.

### *E. Aggregating Throughput by Parallel TCP Connections*

To achieve 1-Mbps stream by one single TCP connection over a long haul connection is challenging. This is due to TCP's throughput that is inversely proportional to RTT, making

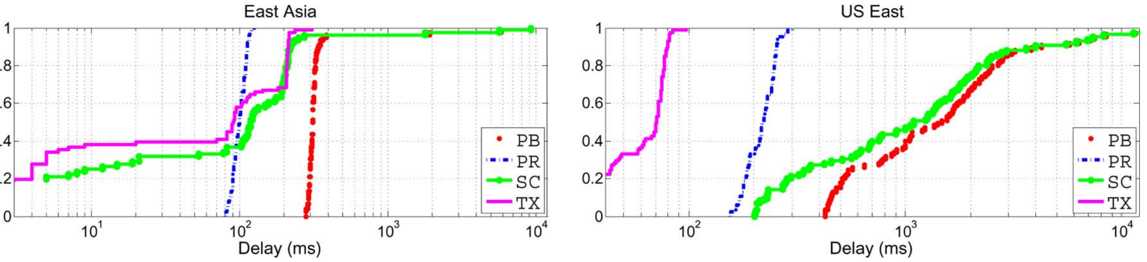


Fig. 12. Decomposing end-to-end playback delay (PB) components: transmission delay (TX), propagation delay (PR), and scheduling delay (SC).

trans-continent connections capped by a rate limit which is usually lower than the 1-Mbps requirement. One way to increase end-to-end throughput is to employ multiple simultaneous TCP connections. In the next experiment, we allow proxies to use parallel TCP connections to communicate with each other. Fig. 10 shows the received streaming rate as we increase the number of parallel connections. We select three locations to demonstrate the benefit. Increasing the number of connections from one to two almost doubles the streaming rate, but the marginal benefit diminishes as more connections are used. An interesting observation is that the maximum achievable streaming rate is around 1.6 Mbps for all locations. Because such a phenomenon is universal all locations, the bottleneck is likely to be the local ISP (HKUST) gateway where a rate limit is applied. This study suggests that parallel TCP connections is often an effective way to achieve high throughput between two proxies, but this approach may be limited due to ISPs' policies.

### F. Scalability of FastMesh-SIM

In our experiment, we successfully provide 1-Mbps streaming to around 100 peers running SIM, with around 20 proxies running FastMesh. It is important to note that the 100 peers, from two campuses and residential users, are supported by 5 proxy nodes. Other proxies, deployed purely for the purpose of studying FastMesh, do not serve these peers directly. A simple back-of-the-envelope calculation projects the system scale to a few hundred peers. In addition, the participating proxies, are normal desktops with stable Internet access such as broadband or campus LAN users (e.g., "super" nodes in the peer-to-peer VoIP), and can be selected based on measurements and history. As such, the two-tier architecture allows the system size to grow proportionally with the number of proxies in FastMesh, which is a promising step towards the goal of scalable 1-Mbps peer-assisted streaming.

## V. Optimizing User Playback Delay

### A. Identifying the Root Cause of Delay Jitters

We first analyze the end-to-end delay jitters. We establish the benchmark performance without enabling the IP-multicast feature and without peer churns. The segment-wise delay log is presented in Fig. 11. We show the result of two peers that are representative of two types of delay jitters, one from East Asia (Korea) that is closer to the server, and the other from US East (Princeton). The East Asia node shows lower mean playback delay and jitters. While occasional hiccups are observed, both nodes are able to achieve an average playback delay of around 500 ms. Since playback delay measures the lifespan of a segment from its birth to its consumption, 500 ms is indeed a challenging delay bound in live streaming. Fig. 11 shows that the US East node suffers more from delay jitters, which is due to the longer trans-continent connection that packets traverse. The next question would be how to identify delay bottlenecks.

### B. Differentiating Fine-Grained Delay Components

To minimize delay, a question that naturally arises is which components of the end-to-end delay are the dominant ones. This allows us to identify bottlenecks in the first place. The peer's playback delay (PB) is decomposed hop by hop, where each hop consists of segment transmission delay (TX), propagation delay (PR), and scheduling delay (SC). To analyze these delay components, we focus on two proxy servers, one located in East Asia (Korea) and the other in US East (Princeton).

Fig. 12 explains the share of different delay components. In this experiment, an end-to-end path that a segment traverses consists of several hops: the hop from the source to proxy servers on the same network, and trans-continental hops between proxy servers. We do not introduce peer churns yet, therefore the additional protocol incurred latency is negligible. In Fig. 12, we show the CDFs of different delay components. The end-to-end delay, propagation delay, and transmission delay are measured, while the scheduling delay is derived by subtracting the other two components from the total end-to-end delay.

For both locations, propagation delay is not constant because different segments may travel along different paths constructed by the FastMesh protocol. For the East Asia node, propagation, transmission, and scheduling delay have approximately an equal share, i.e., 1/3 of the total delay. The US East node shows a larger delay variance than the East Asia node, because the throughput of a longer connection is less stable. In live streaming, we do not skip segments unless a significant lag behind the live playback point is detected. The playback delay is accumulated when a segment arrives late due to the temporary starvation for bandwidth. Therefore, to optimize the user delay, the two bottlenecks should be targeted at: 1) a low-delay mesh constructed using the propagation delay and 2) a bandwidth-sufficient mesh that supports the streaming rate.

The relative magnitude of different delay components also depends on population size, as the total transmission and scheduling delay depends on the scale of our experiments, while the propagation delay does not. We normalize the delay components with respect to our population size, e.g., the tree size in our experiments, and show the projected delay in Table II. As the pop-

TABLE II
NORMALIZED DELAY COMPONENTS OF AN 8-HOP PEER

| Delay Components (ms) | Mean | Percentage | Std |
|---|---|---|---|
| Transmission (TX) | 99 × 7 | 22.6% | 92× 7 |
| Propagation (PR) | 101 | 3.3% | 11× 7 |
| Scheduling (SC) | 325 × 7 | 74.1% | 1216× 7 |
| Playback (PB) | 3069 | 100% | – |

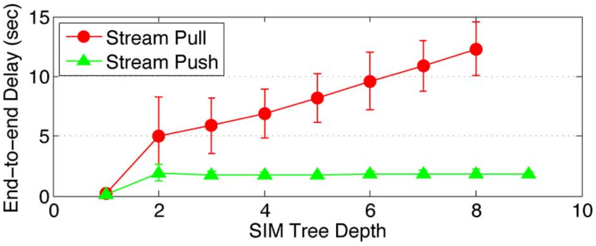


Fig. 13. Push versus pull stream delivery: pull delay grows linearly with tree depth.

ulation size grows, the transmission and scheduling delays tend to increase, which suggests that eliminating protocol overhead should be given a high priority.

### *C. Stream Delivery: Push versus Pull*

As push-based stream delivery is one of our major design decisions, we next quantify how much we can reduce delay compared to a pull-based stream delivery, which is adopted by most commercial applications. We implement a protocol similar to BitTorrent in which peers exchange segment bitmap information and request segments sequentially (compared to video-on-demand which applies a broader set of segment selection policies). We compare a simple pull-based stream delivery, and set a conservative bound on the handshake interval to be 1 s.

Fig. 13 shows the average end-to-end delay of peers that run the SIM protocol from a local campus LAN (Hong Kong). Conventional wisdom is that more tree hops results in higher delay, which is verified by the pull curve, since the protocol (handshake) delay grows proportionally to the tree depth. The data quantifies the intuition that push can significantly reduce delay, where the curve corresponding to "push" is flatter due to the fact that the protocol overhead is no longer the major delay bottleneck as the peer population grows. However, it should be noted that pull-based systems have other potential advantages: better reliability and resource utilization.

### *D. Exploring the Segment Size Tradeoff*

There are various system parameters that one can optimize in any P2P streaming protocol. Here we show the example of optimizing one commonly used parameter, e.g., segment (or so-called chunk) size. Segment size is the minimum exchange unit in peer-to-peer streaming, an important parameter that is employed in both push-based and pull-based systems. A smaller segment size usually reduces per segment transmission time, but at the expense of higher system scheduling overhead. To quantify the best tradeoff, we run experiments that vary the segment size, while keeping other settings fixed. Again we compare two metrics, streaming rate and segment delay, and present the results in Fig. 14.

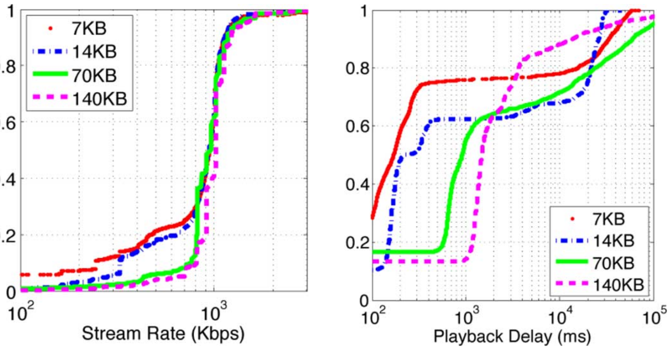


Fig. 14. Experimenting with segment sizes: transmission time versus control overhead tradeoff.

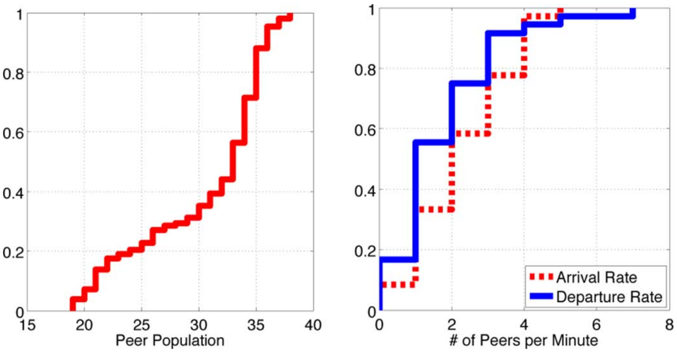


Fig. 15. Introducing heavy node churns to examine system reliability: CDF of system population and peer arrival/departure rates.

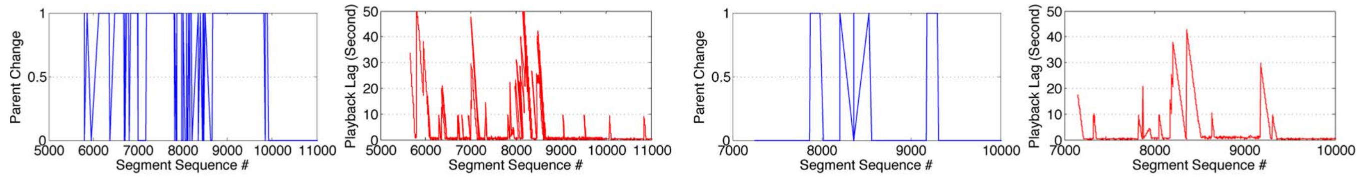
While all trials achieve a mean streaming rate around 1 Mbps, the variance increases monotonically as the segment size decreases. A smaller segment size introduces more control traffic and scheduling overhead, reducing the bandwidth efficiency. The system is potentially more unstable, as shown by those small rates under the setting of small segments. On the flip side, the segment delay is higher for larger segment size as intuition suggests. The median of the delay grows almost linearly with the segment size (although not readily seen from the CDF), which is due to the fact that transmission delay becomes the dominant component as the segment size grows. However, as the segment size decreases, other delay components, e.g., scheduling delay, become more significant. Surprisingly, having a large segment size such as 140 KB reduces worst delays, e.g., greater than 10 s, suggesting that a large segment size is more insensitive to bandwidth variations, though large segment size also increases the average delay. This is also an evidence that scheduling segment retransmission may involve a large system overhead.

## VI. IMPROVING STREAMING RELIABILITY

Maintaining a reliable P2P streaming service is challenging, especially under a high bandwidth demand. There are many factors that can impair the system reliability, among which peer churns and bandwidth fluctuation are the foremost reasons.

### *A. Disruption of Playback Delay Due to Peer Churns*

To understand how churns can affect the system stability and the streaming performance, we let nodes to dynamically leave and rejoin the system. We do not intend to model the churning



(a) (b)

Fig. 16. Performance degradation under peer churns, and demonstration of system recovery from node failovers.

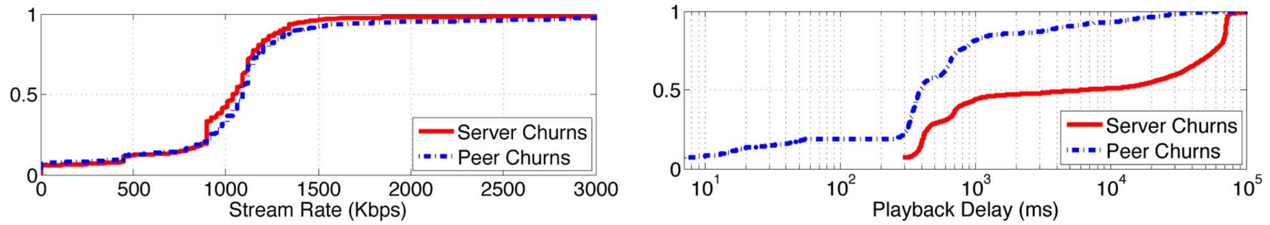


Fig. 17. Comparison of performance under churning peers and proxies: better reliability achieved under peer churns. This motivates an architectural separation of peers and proxies to strike a balance between simplicity and optimality.

behavior of real users; instead, we inject a heavy churn so that it presents a substantial challenge to sustain a smooth streaming experience. For simplicity, we let each node go online and offline repeatedly, and the mean sojourn time for both active and inactive periods is 3 min. For example, Fig. 15 shows the CDF of arrival rates and departure rates that we employ in a light-churn experiment, and the CDF of system population in a trial of one hour long. On average, there are 3 new arrivals and 3 departures in 1 min. In a system with 50 peers, this emulates a strong churning behavior, where both the frequency and percentage of churns are much higher than a normal scenario. Each peer departure incurs tree reconstruction if the peer is an internal node, and each peer arrival may involve several rounds of parent searching until the performance stabilizes.

Fig. 16 shows the segment-wise playback delay of a peer in the system under heavy churns and light churns, respectively. In both Fig. 16(a) and (b), the left curve shows the indicator function of the event that one of the peer's upstream parents has changed, and the right one shows the playback delay of every segment in the same trial. Apparently, the two events, i.e., the parent change and large delay, are highly correlated. Although churns do not penalize the average streaming rate severely, they can greatly impair the stream smoothness by significantly delaying a few segments. In the worst case, a segment may arrive tens of seconds after its live playback point, e.g., when the segment is generated at the source. The playback delay quickly goes back to normal as the tree stabilizes and the required throughput is sustained. In Fig. 16, there are delay spikes when no topology change occurs, which suggests that the bandwidth fluctuation caused by other factors (e.g., cross traffic, packet loss) accounts for the performance degradation.

### B. Server Churns versus Peer Churns

We employ the same set of nodes and run two experiments, one with churning servers and the other with churning peers. Results are shown in Fig. 17. In both cases, the streaming rate exhibits a large variance compared to no-churning experiments. Expectedly peers get stuck when their upstream servers or peers fail, but the protocol can quickly adapt and resume transmission. The suboptimal but adaptive tree protocol is quite robust, despite frequent parent changes. On the other hand, server churns have a more significant impact on the playback delay. This is due to the fact that proxy servers are much fewer than peers, and peers suffer from data starvation when all of their local proxies die. In addition, the FastMesh protocol is not optimized for churns, and the performance penalty under churns is a price we pay for more sophisticated bandwidth aggregation and delay optimization.

### C. What Kind of Trees Are More Robust

Conventional wisdom suggests that large tree fanout (shallow tree) may be, on average, less robust than small tree fanout (deep tree). Large fanout implies higher error correlation among children. The insight we gain from our experiments is indeed a deeper tree is more robust. With proxies and proper error control, a peer's failure does not necessarily trigger the error recovery of all its descendants. The peer's child detects error more quickly than its grandchildren, since the loss of the TCP connection is detected faster than application-layer timeouts.

## VII. Comparing Theory and Practice

There has been a number of papers developing sophisticated algorithms to compute the highest achievable P2P streaming rate, under ideal assumptions such as "no peer churn" and "uplink bandwidth is the only bottleneck" (e.g., [3], [5], [17]). We show that these upper bounds in theory is not much higher than 1 Mbps for the system we measured. This provides another confirmation that achieving 1 Mbps is indeed challenging but still feasible, considering that this measurement study was carried out in a highly stressed environment.

### A. Two-Level Architecture

Many theoretical works [5], [8] proposed a hierarchical architecture in peer-assisted streaming that promotes a separation of the design space into groups of peers clustered by geographic proximity, as illustrated in Fig. 18. Across clusters, super peers, e.g., proxies or peers with high uplink capacities, form a dense core and communicate with one another. Inside each cluster, peers are organized into shallow streaming trees rooted at super

TABLE III
COMPARISON OF THEORETICAL PROPOSAL AND FASTMESH-SIM

| | Theory | FastMesh-SIM |
|---|---|---|
| Inter-Cluster | Cluster heads form rate-optimal mesh | Proxies form mesh with rate-delay tradeoff |
| Intra-Cluster | MutualCast [19]: multiple one or two-hop trees | Shallow trees with IP-multicast support |
| Node Deg. Bound | Yes (# of neighbors) | Yes (# of substreams) |
| Rate Optimality | Inter-cluster: 1/2-approximation; Intra-cluster: optimal | Achieve close-to-optimal performance |
| Algorithm | Centralized | Distributed |

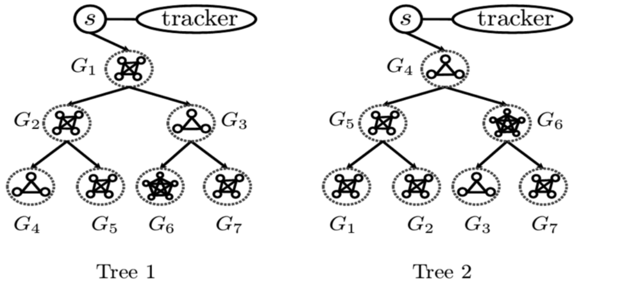


Fig. 18. Two-level hierarchical architecture proposed in [5] that constructs full mesh among proxies in the top level and shallow trees in the bottom level.

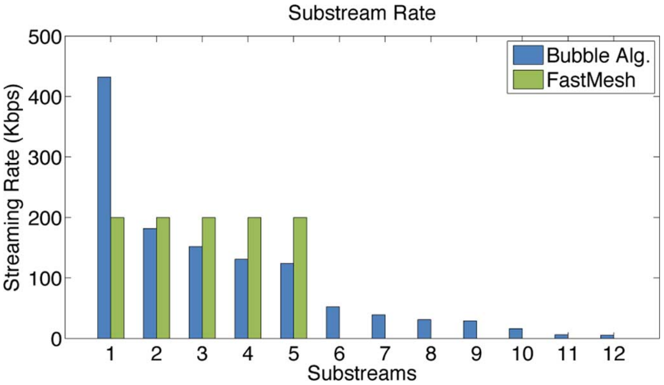


Fig. 19. Comparison of streaming multi-tree construction in FastMesh and Bubble algorithm [5] that achieves optimal streaming capacity under node degree bounds.

peers. The two-level architecture is scalable and of low complexity, and ensures high bit-rate streaming is sustainable over a wide geographic coverage.

Table III compares and contrasts theoretical suggestions and the practical design in FastMesh-SIM. While FastMesh-SIM and P2P streaming capacity work were independently carried out, it turns out that FastMesh-SIM captures these architectural insights from the theory work and confirms them with a global-scale deployment and measurement. Later in this section, we show the similarity of streaming tree constructions, as well as proximity of theoretical performance bounds and what is achieved in practice.

### *B. Similarity of Streaming Tree Construction Methods in Theory and Practice*

We later show that FastMesh-SIM achieves a very large fraction of the streaming rate upper-bound. To first provide an intuition why it works well in practice, we compare the resulting streaming trees constructed in FastMesh protocol and one of our recent studies in P2P streaming capacity under node degree bounds [5]. Following step-by-step the algorithm in FastMesh and Bubble algorithm, i.e., the multi-tree construction algorithm in [5], we calculate individual substream rate, i.e., the bit-rate supported on each streaming tree, under the same peer capacities measured from the dataset shown in the full paper [20, Appendix]. Results are illustrated in Fig. 19.

Bubble algorithm is executed under a degree bound of 4 (that approximates $\log N$ for scalability). Bubble algorithm produces a total of 12 streaming trees, with corresponding substream rates ranging from 5 kbps to 430 kbps. In contrast, FastMesh protocol explicitly tries to construct 5 streaming trees and each substream delivers 200 kbps bit-rate (which is a tunable parameter in our protocol given the total target streaming rate is 1 Mbps). The similarity between the multi-tree constructed by theory and FastMesh-SIM is obvious: they both capture the set of major streaming trees that can support high bit-rates. Although the FastMesh protocol uses low-complexity heuristics, it is able to find "good" trees out of an exponential number of possibilities. FashMesh does lose substream rates in the long tail that contribute to streaming rate optimality, but comes with lower implementation complexity and better system reliability.

TABLE IV
THEORY-PRACTICE GAP IN P2P STREAMING CAPACITY: A SURPRISINGLY LARGE FRACTION OF THEORY UPPER-BOUND IS ATTAINABLE BY FASTMESH-SIM, CONFIRMING THE ACCURACY OF THEORETICAL MODELS

| Rate (kbps) | Theory upper bound | Practice average |
|---|---|---|
| Unrestricted | 1123 | 929 |
| Degree bounded | 1062 [3] | |
| With helper | 1201 [5] | 1021 |

### *C. FastMesh-SIM Achieves Close-to-Optimal Streaming Rate*

We use simplified versions of the state-of-the-art algorithms to derive an upper bound on streaming rate through back-of-the-envelope calculation, presented in [20, Appendix]. Such a simplification further loosens the ideal bounds, and the resulting upper bounds are even more optimistic. The input parameters to the upper bound calculation is the set of peer uplink capacities measured in the trace, and the upper bound is computed using the 95-percentile of those across all the peers. Results are given in Table IV. The resulting upper bound on streaming capacity is 1.201 Mbps while in practice we achieved 1.021 Mbps on average. This is a surprisingly large fraction, over 85%, of the upper bound generated by following many idealistic assumptions in theory.

## VIII. RELATED WORK

The widespread deployment of P2P streaming systems has motivated many types of work on the understanding of challenges of streaming video content over the public Internet.

*1) Architecture of Peer-Assisted Live Streaming:* Similar to our work, some systems [21], [22] implement a hybrid approach by leveraging the best of server infrastructures (e.g., CDNs) and peers. The benefits of utilizing peer resources to distribute contents are studied in [23]. Another deployment [2] of hybrid P2P-CDN system also confirms such a design choice. Other work [24] also studied the feasibility of building a peer-assisted high-quality VoD system. Most of these works are either trace-based simulations leveraging data collected from CDNs and BitTorrent, or deployed in a smaller region under the conventional low-quality streaming, and thus, lack the validation as a global 1-Mbps streaming by an operational system.

*2) Measurements of Commercial P2P Streaming Systems:* A number of measurement works studied commercial P2P streaming applications [7], [8], [11], [25], [26], characterizing the performance in a large user population, and revealing the challenges faced by today's operational systems. These works passively sniff, or actively crawl the online users to collect and infer various system performance metrics. With similar goals, [12], [27], [28], [29], leverage data contributed from service providers, inspect current system design, reveal the viewing behaviors of real users, and characterize large-scale topological dynamics. This approach, complementary to this paper's, is highly valuable in providing a better understanding of how popular applications, and real users, behave in large commercial deployments.

A smaller scale and less mature version of our implementation involving only Hong Kong and Princeton peers was tested and reported in a short paper [30]. We experiment with lower streaming rates of only 300 kbps, as is typical in most of today's commercial systems. These earlier experiments lacked most of the key features examined, metrics measured, and lessons learned in the experiments that achieved the substantially higher streaming rate in the current paper.

## IX. Conclusion

In the tradeoff space between high visibility/control, and a high degree of realism, this paper presents a complementary angle to existing approaches. We design controlled experiments with target research questions in mind, and analyze fine-grain measurement data collected from the highly configurable FastMesh-SIM platform. With a global footprint over 8 counties, our experiments demonstrate that 1-Mbps streaming over the global Internet is indeed an attainable goal, and, as indicated by the back-of-the-envelope calculation of theoretical upper bound benchmarks, it is not an easy one to achieve. The collective use of design choices, ranging from architectural decisions, to parameter selection, provides quantitative understanding on how high bit-rate streaming can be achieved, and useful lessons to building a global 1-Mbps streaming service.

## Acknowledgment

The authors would like to thank colleagues around the world who helped host and maintain the nodes of our experiments in their countries, including L. Andrew in Australia, K. Leung in the U.K., S. Low in the USA, F. Paganini in Uruguay, and Y. Yi in Korea.

## References

[1] PPLive. [Online]. Available: http://www.pplive.com/.

[2] H. Yin, X. Liu, T. Zhan, V. Sekar, F. Qiu, C. Lin, H. Zhang, and B. Li, "Design and deployment of a hybrid CDN-P2P system for live video streaming: Experiences with livesky," in *Proc. ACM Multimedia*, 2009.

[3] S. Liu, R. Zhang-Shen, W. Jiang, J. Rexford, and M. Chiang, "Performance bounds for peer-assisted live streaming," in *Proc. ACM SIGMETRICS*, 2008.

[4] X. Jin, K.-L. Cheng, and S.-H. Chan, "Scalable island multicast for peer-to-peer streaming," *J. Adv. Multimedia, Special Issue on Multimedia Networking*, 2007.

[5] S. Liu, M. Chen, S. Sengupta, M. Chiang, J. Li, and P. A. Chou, "P2P streaming capacity under node degree bound," in *Proc. Int. Conf. Distributed Computing Systems*, 2010.

[6] S. Sengupta, S. Liu, M. Chen, M. Chiang, J. Li, and P. A. Chou, "Peer-to-Peer streaming capacity," *IEEE Trans. Inf. Theory*, vol. 57, no. 8, pp. 5072–5087, Aug. 2011.

[7] S. Ali, A. Mathur, and H. Zhang, "Measurement of commercial peer-to-peer live video streaming," in *Proc. Workshop Recent Advances in Peer-to-Peer Streaming (WRAIPS)*, 2006.

[8] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1672–1687, Dec. 2007.

[9] X. Hei, Y. Liu, and K. W. Ross, "Inferring network-wide quality in P2P live streaming systems," *IEEE J. Select. Areas Commun.*, vol. 25, no. 9, pp. 1640–1654, 2007.

[10] C. Wu, B. Li, and S. Zhao, "Characterizing peer-to-peer streaming flows," *IEEE J. Select. Areas Commun.*, vol. 25, no. 9, pp. 1612–1626, Dec. 2007.

[11] E. Alessandria, M. Gallo, E. Leonardi, M. Mellia, and M. Meo, "P2P-TV systems under adverse network conditions: A measurement study," in *Proc. IEEE INFOCOM*, 2009.

[12] C. Wu, B. Li, and S. Zhao, "Exploring large-scale peer-to-peer live streaming topologies," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 4, no. 3, pp. 1–23, 2008.

[13] X. Jin, H.-S. Tang, S.-H. Chan, and K.-L. Cheng, "Deployment issues in scalable island multicast for peer-to-peer streaming," *IEEE Multimedia Mag.*, vol. 16, no. 1, pp. 72–80, Jan.–Mar. 2009.

[14] D.-N. Ren, Y.-T. H. Li, and S.-H. Chan, "FastMesh: On reducing mesh delay for peer-to-peer live streaming," in *Proc. IEEE INFOCOM*, 2008.

[15] S. Zhang, Z. Shao, and M. Chen, "Optimal distributed P2P streaming under node degree bounds," in *Proc. Int. Conf. Network Protocols*, 2010.

[16] W.-P. Yiu, K.-F. Wong, S.-H. Chan, W.-C. Wong, Q. Zhang, W.-W. Zhu, and Y.-Q. Zhang, "Lateral error recovery for media streaming in application-level multicast," *IEEE Trans. Multimedia, Special Issue on Distributed Media Technologies and Applications*, vol. 8, no. 2, pp. 219–232, Apr. 2006.

[17] Y. Liu, "On the minimum delay peer-to-peer video streaming: How realtime can it be?," in *Proc. ACM Multimedia*, 2007.

[18] Y. Liu, "Delay bounds of chunk-based peer-to-peer video streaming," *IEEE/ACM Trans. Networking*, vol. 18, no. 4, pp. 1195–1206, Aug. 2009.

[19] J. Li, P. A. Chou, and C. Zhang, "Mutualcast: An efficient mechanism for one-to-many content distribution," in *Proc. ACM SIGCOMM ASIA Workshop*, 2005.

[20] J. W. Jiang, S.-H. G. Chan, M. Chiang, J. Rexford, D. T. Ren, and B. Wei, *Global 1 Mbps Peer-Assisted Streaming: Fine-Grain Measurement of a Configurable Platform*. Princeton, NJ: Princeton Univ. Tech. Rep.

[21] C. Huang, A. Wang, J. Li, and K. W. Ross, "Understanding hybrid CDN-P2P: Why limelight needs its own red swoosh," in *Proc. NOSSDAV*, 2008.

[22] D. Xu, S. S. Kulkarni, C. Rosenberg, and H. K. Chai, "A CDN-P2P hybrid architecture for cost-effective streaming media distribution," *Comput. Netw.*, vol. 44, pp. 353–382, 2004.

[23] T. Karagiannis, P. Rodriguez, and K. Papagiannaki, "Should Internet service providers fear peer-assisted content distribution?," in *Proc. Internet Measurement Conf.*, 2005.

[24] S. Annapureddy, S. Guha, C. Gkantsidis, D. Gunawardena, and P. R. Rodriguez, "Is high-quality VOD feasible using P2P swarming?," in *Proc. World Wide Web*, 2007, pp. 903–912.

[25] T. Silverston and O. Fourmaux, "P2P IPTV measurement: A comparison study," *CoRR*, 2006.

[26] F. Wang, J. Liu, and Y. Xiong, "Stable peers: Existence, importance, and application in peer-to-peer live video streaming," in *Proc. IEEE INFOCOM*, 2008.

[27] M. Cha, P. Rodriguez, J. Crowcroft, S. Moon, and X. Amatriain, "Watching television over an IP network," in *Proc. Internet Measurement Conf.*, 2008.

[28] Y. Huang, T. Z. Fu, D.-M. Chiu, J. C. Lui, and C. Huang, "Challenges, design and analysis of a large-scale P2P-VOD system," in *Proc. ACM SIGCOMM*, 2008.

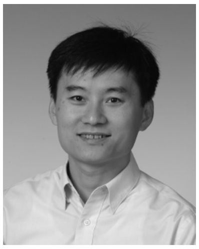
[29] S. Xie, B. Li, G. Keung, and X. Zhang, "Coolstreaming: Design, theory, and practice," *IEEE Trans. Multimedia*, vol. 9, no. 8, pp. 1661–1671, Dec. 2007.

[30] J. W. Jiang, S.-H. Chan, M. Chiang, J. Rexford, K.-F. S. Wong, and C.-H. P. Yuen, "Proxy-P2P streaming under the microscope: Fine-grain measurement of a configurable platform," in *Proc. Int. Conf. Computer Communication Networks (ICCCN)*, 2010.

**Joe Wenjie Jiang** received B.Sc. degree from the University of Science and Technology of China in 2003, the M.Phil. degree from the Chinese University of Hong Kong in 2005, and the Ph.D. degree in computer science from Princeton University, Princeton, NJ, in 2012.

He was coadvised by Prof. J. Rexford and Prof. M. Chiang. His thesis title was "Wide-area traffic management for cloud services." His research interests include content distribution and cloud services, data center traffic management, Internet routing, and video streaming.

Dr. Jiang served on the Program Committee for ACM S3 2011 (collocated with MOBICOM 2011), and was Co-Chair of CCNC 2010 Student Workshop. His research received 2005 Performance Best Student Paper Award, and 2005 Best Research Output by Research Postgraduate Students from the Chinese University of Hong Kong.



**S.-H. Gary Chan** (S'89–M'98–SM'03) received the B.S.E. degree (highest honor) in electrical engineering from Princeton University, Princeton, NJ, in 1993, with certificates in applied and computational mathematics, engineering physics, and engineering and management systems. He received the M.S.E. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, in 1994 and 1999, respectively, with a minor in business administration.

He is currently an Associate Professor of the Department of Computer Science and Engineering, Director of Sino Software Research Institute, and Co-Director of Risk Management and Business Intelligence program, The Hong Kong University of Science and Technology (HKUST), Hong Kong. His research interest includes multimedia networking, peer-to-peer streaming and technologies, and wireless communication networks. He has been a Visiting Professor/Adjunct Researcher in Microsoft Research Asia (2000–2011), Research Collaborator at Princeton University (2009), Visiting Associate Professor at Stanford University (2008–2009), Director of Computer Engineering Program at the HKUST (2006–2008), Visiting Assistant Professor in Networking at the University of California at Davis (1998–1999), and Research Intern at the NEC Research Institute, Princeton, NJ (1992–1993).

Dr. Chan has been an Associate Editor of the IEEE Transactions on Multimedia (2006–2011), and is a Vice-Chair of the Peer-to-Peer Networking and Communications Technical Sub-Committee of the IEEE Comsoc Emerging Technologies Committee. He has been Guest Editors of IEEE Transactions on Multimedia (2011), IEEE Signal Processing Magazine (2011), IEEE Communication Magazine (2007), and *Springer Multimedia Tools and Applications* (2007). He was the TPC chair of the IEEE Consumer Communications and Networking Conference (CCNC) 2010, Multimedia symposium in IEEE Globecom (2007 and 2006) and IEEE ICC (2007 and 2005), and Workshop on Advances in Peer-to-Peer Multimedia Streaming in ACM Multimedia Conference (2005). He is the recipient of the Google Mobile 2014 award in 2010 and 2011, and is a member of honor societies Tau Beta Pi, Sigma Xi and Phi Beta Kappa. He was a William and Leila Fellow at Stanford University (1993–1994). At Princeton, he was the 1993 recipient of the Charles Ira Young Memorial Tablet and Medal and the POEM Newport Award of Excellence.

**Mung Chiang** (S'00–M'03–SM'08–F'12) received the B.S. (Hons.), M.S., and Ph.D. degrees from Stanford University, Stanford, CA, in 1999, 2000, and 2003, respectively

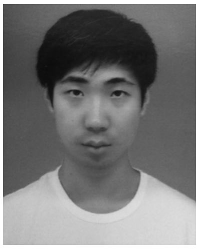
He is a Professor of electrical engineering at Princeton University, Princeton, NJ, and an affiliated faculty in applied and computational mathematics, and in computer science. He was an Assistant Professor 2003–2008 and an Associate Professor 2008–2011 at Princeton University.

Dr. Chiang's research on networking received the 2012 IEEE Kiyo Tomiyasu Award, a 2008 U.S. Presidential Early Career Award for Scientists and Engineers, several young investigator awards, and a few paper awards, including the 2012 IEEE INFOCOM Best Paper Award. His inventions resulted in a few technology transfers to commercial adoption, and he received a 2007 Technology Review TR35 Award and founded the Princeton EDGE Lab in 2009. He served as an IEEE Communications Society Distinguished Lecturer in 2012–2013, and wrote an undergraduate textbook: *Networked Life: 20 Questions and Answers* (Cambridge, U.K.: Cambridge Univ. Press, 2012).

**Jennifer Rexford** received the B.S.E. degree in electrical engineering from Princeton University, Princeton, NJ, in 1991, and the M.S.E. and Ph.D. degrees in computer science and electrical engineering from the University of Michigan, Ann Arbor, in 1993 and 1996, respectively.

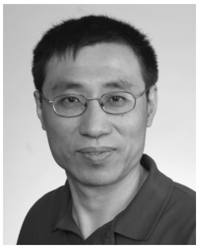
She is a Professor in the Computer Science Department at Princeton University. From 1996–2004, she was a member of the Network Management and Performance Department at AT&T Labs-Research. She is co-author of the book *Web Protocols and Practice* (Reading, MA: Addison-Wesley, 2001).

Dr. Rexford served as the chair of ACM SIGCOMM from 2003 to 2007. She was the 2004 winner of ACM's Grace Murray Hopper Award for outstanding young computer professionals.



**D. Tony Ren** received the B.Eng. degree in computer science (information engineering) from the Hong Kong University of Science and Technology (HKUST) in 2007, and the M.Phil. degree in computer science from HKUST in 2009. He is currently pursuing the Ph.D. degree at the Department of Computer Science and Engineering in HKUST, supervised by Prof. S. H. G. Chan.

His research interest includes live streaming technologies, multimedia networking, overlay, and peer-to-peer networks.



**Bin Wei** received the Ph.D. degree in computer science from Princeton University, Princeton, NJ.

He is a research staff member at AT&T Labs Research, Florham Park, NJ. His contributions to multimedia communications focus on the middleware which provides multimedia services for various user devices ranging from display walls to handheld devices by building prototype systems. He also works on improving communication performance and user experience with mobile devices. He has many publications in major international technical conferences and journals.