2007

Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective

Jeff Seibert

Davied Zae

Sonia Fahmy
Purdue University, fahmy@cs.purdue.edu

Cristina Nita-Rotaru
Purdue University, crisn@cs.purdue.edu

Report Number:
07-020

EXPERIMENTAL COMPARISON OF PEER-TO-PEER STREAMING OVERLAYS: AN APPLICATION PERSPECTIVE

Jeff Seibert
David Zage
Sonia Fahmy
Cristina Nita-Rotaru

Department of Computer Science
Purdue University
West Lafayette, IN 47907

CSD TR #07-020
July 2007

Experimental Comparison of Peer-to-Peer Streaming Overlays: An Application Perspective

Jeff Seibert, David Zage, Sonia Fahmy, Cristina Nita-Rotaru
Department of Computer Science, Purdue University
E-mail: {jcseiber, zagedj, fahmy, crisn}@cs.purdue.edu

Abstract—**Peer-to-peer streaming systems are becoming highly popular for IP Television (IPTV). Most systems can be categorized as either tree-based or mesh-based, and as either push-based or pull-based. However, there is a lack of clear understanding of how these different mechanisms perform comparatively in a real-world setting. In this paper, we compare two representative streaming systems using mesh-based and multiple tree-based overlay routing through deployments on the PlanetLab wide-area experimentation platform. To the best of our knowledge, this is the first study to directly compare streaming overlay architectures in real Internet settings. Our results indicate that mesh-based systems inject a much higher number of duplicate packets into the network, but they perform better under a variety of conditions. In particular, mesh-based systems give consistently higher application goodput when the number of overlay nodes, or the streaming rates increase. They also perform better under churn and large flash crowds. Their performance suffers when latencies among peers are high, however. Overall, mesh-based systems appear to be a better choice than multi-tree based systems for peer-to-peer streaming at a large scale.**

I. Introduction

In recent years, there has been an explosion of interest in peer-to-peer (P2P) streaming of audio and video in real-time [1], [2], [3], [4], [5]. Most streaming systems for IP Television (IPTV) utilize an overlay (application-level) multicast group where peers receiving a stream can serve as proxies that forward content to other peers. A number of studies have suggested and implemented a variety of overlay designs [6], [7], [8], [9], [10] to meet the stringent demands of the commercial market, replacing network-layer (IP) multicast.

Internet streaming has different requirements [11] from other P2P applications (e.g., file sharing, video-on-demand), making the design of overlay networks for such applications a challenging task. Streaming imposes stringent real-time requirements on throughput and latency. Specifically, streaming applications must sustain throughputs that ensure high quality of video and audio, while providing simultaneous support for a large number of participants with dynamic changes in group membership. In addition, data has to meet deadlines to ensure smooth playback of the content in real-time.

Two architectures for Internet streaming have emerged in recent years: tree-based and mesh-based architectures. A tree-based streaming overlay constructs a tree where the source broadcasting the stream is the root of the tree and every other peer in the network is a child of either the source or another peer. Data simply flows down the tree to all participating members. An example of a tree-based multicast overlay is ESM [12]. However, in a single tree-based routing topology, leaf nodes do not forward data, leading to an imbalance in the load on the peers. To address this limitation, recent research has introduced multi-tree overlays. Such systems distribute bandwidth costs across participants by disseminating the data on multiple dissimilar trees. Examples of multi-tree multicast overlays are Chunkyspread [8] and SplitStream [6].

A mesh-based streaming overlay facilitates data dissemination in a less structured manner, by requiring peers to exchange data with a subset of the nodes in the network, maintained in the form of a neighbor set. The major difference from tree-based overlays is that in mesh-based systems, there is no predefined route in which data flows. Examples of mesh-based multicast overlays are Chainsaw [7] and CoolStreaming/DONet [9]. Several highly popular IPTV systems, such as PPLive [4] and PPStream [5], also extend ideas from the mesh-based BitTorrent [13] for real-time streaming. Meshes are characteristically resilient to churn and node failures, but exhibit high overhead.

While several design variants have been proposed for tree-based and mesh-based overlays, there is a lack of clear understanding of which designs perform better in a real-world setting. A concrete characterization of the conditions under which each provides a better service *to the application* is missing. Previous studies have compared overlay multicast networks via simulations and limited Internet experiments, including [14], [15], [16], [17], but none focused on streaming applications. One exception is the work in [18] which conducts a simulation comparison of a multi-tree system similar to SplitStream and PRIME [19] – a recently proposed overlay multicast system which combines the advantages of mesh and tree designs.

In this paper, we evaluate two representative systems through deployments on the PlanetLab wide-area experimentation platform [20]. We select the mesh-based Chainsaw [7] and the tree-based SplitStream [6] systems because their core designs are based on a basic bidirectional mesh or a basic multi-tree topology, and their implementations are publicly available. To the best of our knowledge, this is the first study to directly compare streaming overlay architectures in real Internet settings. We identify the pros and cons of mesh-based and multi-tree based overlay multicast networks with respect to P2P streaming under a variety of conditions. Our study considers not only intuitive aspects such as scalability and performance under churn, but also less studied factors such as

bandwidth and latency heterogeneity of overlay participants.

The remainder of this paper is organized as follows. Section II classifies overlay multicast streaming systems. Section III discusses our criteria for comparison and describes the two systems we compare. Section IV gives our experimental methodology and results. Section V surveys related work. Finally, section VI summarizes the conclusions from our study.

II. Taxonomy of Overlay Multicast Approaches

The earliest overlay multicast systems used a single-tree topology, and did not specifically address real time streaming requirements [12], [21], [22], [23]. For example, Overcast [22] was designed for reliable communication, such as file distribution. Later, some of the overlay multicast systems were extended for the Internet streaming application; for example, ESM [12] was extended and deployed for streaming as discussed in [24]. Multi-tree systems such as CoopNet [25], SplitStream [6], and Chunkyspread [8] were later proposed.

Mesh-based systems, e.g., CoolStreaming/DONet [9], and Chainsaw [7], were proposed to address the inherent lack of resilience of tree-based structures. Hybrid systems such as Bullet [10] and mTreebone [26] have also been proposed: these utilize a tree to initially send data and then use a mesh to send the data that each node is missing. The first pure meshes used *bidirectional* links to send data back and forth between neighbors. Later, mesh-based systems such as MeshCast [27] and PRIME [19] used links unidirectionally, separating peers into either sender or receiver groups.

We can also categorize overlays into push- or pull-based systems. Characteristically, tree-based overlays are push-based: every parent will automatically send all the data it receives to each of its children without them requesting it. Meshes are typically pull-based: participants must request packets from their neighbors. This affects the control message overhead required by each type of overlay. Push-based systems typically exhibit lower overhead since they simply need to maintain the overlay structure. Pull-based systems need to continuously update peers concerning what parts of the stream each node has, thus creating high control overhead.

TABLE I
Classification of Overlay Multicast Systems

System	Peer Discovery	Topology		Push/Pull
ESM	Underlying mesh	Single	Tree	Push
Overcast	Source			
NICE	Bootstrap node			
SplitStream	Pastry	Multiple		
CoopNet	Source			
Chunkyspread	SwapLinks			
mTreebone	Source	Tree+mesh		Both
Bullet	RanSub			
MeshCast	Bootstrap node	Unidirectional	Mesh	Pull
PRIME	Bootstrap node			
CoolStreaming	Peers	Bidirectional		
Chainsaw	Bootstrap node			

Table I classifies a set of popular overlay multicast approaches according to the mechanisms they employ. In the table, "peer discovery" refers to how each node finds new neighbors once after it has joined the overlay. RanSub [28] and SwapLinks [29] are distributed algorithms that find nodes to peer with. Based on this classification, we have selected Chainsaw [7] and SplitStream [6] for our experiments, because their core design reflects a basic bidirectional mesh or multi-tree topology.

III. Comparing P2P Streaming Approaches

In this section, we discuss the two systems we have selected for our comparison study, and present the criteria by which we compare them.

A. Chainsaw

Chainsaw [7] is a single-source, multiple-receiver, mesh-based overlay utilizing a pull-based approach in which nodes request packets from a set of peer nodes, referred to as the *neighbor set*. A new node obtains this set at join time by contacting a bootstrap node. A node attempts to maintain a minimum number of neighbors; if a peer disconnects, the node requests more peers from the bootstrap node. Nodes never refuse a connection request from any peer.

Whenever a node receives a new packet, it notifies its neighbors about it. In addition, each node maintains information about packets available for other peers, referred to as *window of availability*, i.e., a buffer that contains packets that have recently been received and about which peers were notified. Packets are discarded after a certain amount of time to prevent old data from being propagated in the overlay.

Each node also maintains a list of the packets it is interested in, referred to as *window of interest*, by tracking the notifications of available packets advertised by each of its neighbors that it does not have. Based on the window of interest, a node randomly selects packets to request from all available peers. Each node requests packets from different neighbors to minimize the number of missed packets.

B. SplitStream

SplitStream [6] is a single-source, multiple-receiver, multi-tree overlay utilizing a push-based approach in which the source disseminates data over several disjoint trees. Since the root and all the other interior nodes will, if possible, be different for every tree, the bandwidth cost of relaying data is distributed among all participants. The trees are constructed using Scribe [30], an application level multicast infrastructure that is itself built on top of the Pastry Distributed Hash Table (DHT) [31].

To join, a node contacts a bootstrap node that may not necessarily be the source. Once a node is part of the overlay, it subscribes to each tree from which it wishes to receive content. A node can explicitly declare the maximum number of children that it wishes to support. Each node maintains information about each tree that it is part of, i.e., the identity of its parent and children. A node forwards all packets on to each of its children, assuming it is an interior node for the tree which these packets were sent on. The source splits the

stream into packets and then sends the data down each tree. SplitStream does not adapt its trees unless a node fails or quits the overlay.

C. Comparison Criteria

In general, P2P systems deployed over the Internet are expected to scale well with the number of participants and take advantage of the diverse resources contributed by each participant. In addition, Internet *streaming applications* have specific characteristics that place additional requirements on P2P streaming overlays. They must be able to sustain bandwidths in the range of 300 kbps to 1 Mbps [32], with 1 Mbps delivering "TV quality" audio and video [33], and be able to provide uninterrupted service in the presence of churn and flash crowds. They must also deliver data within a given time, usually on the order of a few seconds, to ensure smooth playback of video. As a result, data that arrives late is not useful for the application and unnecessarily consumes bandwidth. To ensure that these requirements are met, streaming overlays often duplicate data in the network, resulting in traffic which may not be useful from the application perspective (in addition to control messages sent to maintain information about the overlay structure and required data).

Based on these observations, our comparison examines the following aspects that are crucial from an application perspective:

(1) Scalability with application-prescribed streaming rates: Obviously, the higher the bandwidth, the higher the quality of the streaming video provided to the application. We study the degree to which mesh and multi-tree based overlays can sustain bandwidths needed or expected to be needed in the future, seeking to identify any possible saturation points.

(2) Scalability with the number of overlay participants: We investigate how well mesh and multi-tree based overlays scale with increasing number of participants.

(3) Unusable data: Since streaming video over the Internet requires stringent deadlines to be met, only data received before each deadline is useful. Unusable data therefore includes both duplicates and data that arrived too late to be relevant. Usable data constitutes the application goodput.

(4) Impact of bandwidth heterogeneity of overlay participants on system performance: Streaming overlays must be able to operate under the diversified bandwidth capabilities of users over the Internet. We examine which overlay strategy better exploits this diversity and does not penalize nodes with low-bandwidth connections.

(5) Impact of latency heterogeneity of overlay participants on system performance: Similar to the diversified bandwidth capabilities, nodes also exhibit a diversified range of latencies to other peers and to the broadcast source. We investigate how the overlays we compare perform in a setting with nodes having a mix of latency values.

(6) System recovery when confronted with flash crowds: First experienced in web-based applications, flash crowds were shown to occur frequently in Internet streaming [34]. Hence, an overlay must be able to quickly integrate newcomers into the overlay and ensure a small startup delay.

(7) System performance under high churn: Peers leaving the system during a given period can adversely affect the performance of the system, as some nodes may find themselves disconnected or experience a temporary service interruption. We investigate the performance of mesh and multi-tree based overlays under high churn.

IV. Experimental Results

In this section, we directly compare the two overlay systems, Chainsaw and SplitStream, using real-world deployments and metrics derived from the goals stated in Section III-C.

A. Experimental Methodology

To study the two systems under real-world conditions, we conducted our experiments on PlanetLab [20]. PlanetLab provides a research platform for large scale distributed experimentation of peer-to-peer systems over the Internet [35]. In order to mitigate the possible limitations of using a testbed, such as those addressed in [35], we ran several experiments at different times of the day and different days of the week and computed the variance of our results. As can be seen from Figure 1, there is little variability in the systems with respect to the time of day the experiments were performed. Further, we randomly selected experimental nodes for different experiments (subject to certain constraints as discussed later in this section) to validate the statistical significance of results, and nodes were chosen to span multiple operational and administrative domains. Each experiment was repeated ten times.

We used streaming bit rates of 400 kbps to 1 Mbps, which are representative of the bit rates used in many current video streaming applications [32]. The source was always located on a host at Purdue University. We configured the source to wait for 30 seconds before starting to send data. We consider that a packet must arrive within 5 seconds to be considered useful, according to the buffer times used in [36], [37], [38][1]. We used a maximum of 280 nodes in our experiments because that is the largest number of nodes with access bandwidth greater than 1 Mbps that we could connect to.

We configured Chainsaw such that each node uses a minimum of 15 neighbors, and assumes the request for a packet is lost after 1 second. The source connects to twice as many neighbors as a regular node and pushes two copies of every packet. We configured SplitStream to use 16 trees, with every node joining all trees. A node sends every packet to every child (assuming that it is a packet for that tree). These Chainsaw and SplitStream parameters are the same as in [39] and [6] respectively. We used a default packet size of 2500 bytes for Chainsaw unless otherwise specified. For SplitStream, since one packet per second was sent through each tree, the packet size was determined by the desired streaming rate, and changes per experiment.

[1]Our experiments with 10 and 15 second thresholds revealed that both systems perform only marginally better.

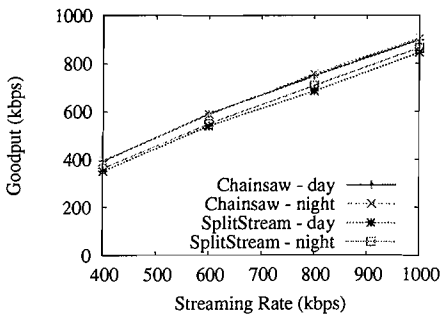


Fig. 1. Goodput at different times of the day and different streaming rates using a configuration of 112 nodes that have over 1 Mbps bandwidth capabilities

We compare the two systems by evaluating the following metrics:

- **Goodput** is the average rate of data that was received before the deadline (5 seconds), and that had not been received before.
- **Late Data** is the average rate of data that was received after the deadline.
- **Duplicate Data** is the average rate of data that was received before the deadline, but that had been received before.
- **Throughput** is the average rate at which all application data is received. In other words, $Throughput = Goodput + LateData + DuplicateData$.
- **Continuity Index**, defined by Pai *et al.* [7], is used to measure the effect of churn. It is equal to the goodput divided by the total amount of data that could have possibly been received while a peer participated in the overlay. This is equivalent to $\frac{Goodput}{StreamingRate}$.

B. Scalability with Streaming Rates

To compare how well each overlay scales with increasing streaming rates, we vary the streaming rate from 400 kbps to 1 Mbps, using a deployment of about 280 nodes for both Chainsaw and SplitStream. In each experiment, the source streamed data for 20 minutes. We used all responsive nodes on PlanetLab that had high access bandwidth (greater than 1 Mbps) and low latency (with average Round Trip Times (RTTs) of 100 ms to the source). Figure 2 shows the results and the means with 95% confidence intervals.

Figure 2(a) depicts the average throughput of all nodes. In an ideal case, the application data received would be identical to the streaming rate. It can be seen that the throughput used by Chainsaw is very close to the ideal. In contrast, SplitStream receives considerably less than the ideal, especially as the streaming rate increases. As seen in Figure 2(b), the goodput for both overlays is less than the streaming rate, with SplitStream suffering more for higher streaming rates. The confidence intervals depicted on the figures are considerably wider for SplitStream than for Chainsaw, demonstrating that SplitStream performance has a higher variability across the ten identical experiments. SplitStream is thus more sensitive to Internet conditions.

The reason for the low goodput of SplitStream is depicted in Figure 2(c). SplitStream receives a non-negligible amount of late data (data received after the 5 second deadline) – higher than the late data received by Chainsaw. We attribute this to the fact that Chainsaw is a pull-based system, where each peer decides what pieces of information it needs. Unlike Chainsaw, SplitStream uses a push-based approach in which nodes push data to their children on different trees at different times. This, combined with the lack of any mechanism for dropping late data, results in unnecessary bandwidth consumption.

Figure 2(d) shows the duplicate data for both overlays. SplitStream, being tree-based, receives a negligible amount of duplicate data, whereas Chainsaw suffers from a slightly growing amount of duplicate data as streaming rates increase. SplitStream received negligible amounts of duplicate data in all the experiments presented in this paper.

In summary, our results demonstrate that Chainsaw outperforms SplitStream at higher streaming rates. Surprisingly, in the range of 400 kbps to 1 Mbps, we found no saturation point, meaning that for our 280 node scenario, neither system has an inherent streaming rate below 1 Mbps where it cannot send any more data.

C. Scalability with Overlay Size

Figure 3 demonstrates the impact of the size of the multicast group when using a streaming rate of 1 Mbps for 20 minutes, varying the number of overlay nodes from 80 to 280 (the number of responsive PlanetLab nodes with good bandwidth and latency properties). We also repeated the set of experiments for a streaming rate of 500 kbps and the results were consistent (with SplitStream and Chainsaw being closer in performance). We omit these results due to space limitations.

As the number of nodes participating in the overlay increases, we can see from Figure 3(b) that the goodput of Chainsaw slightly increases, without a corresponding increase in throughput (Figure 3(a)). This demonstrates that Chainsaw scales with the number of participants in the overlay and is able to effectively use the available resources without increasing the amount of late or duplicate data in the system. However, this is not the case with SplitStream. Although SplitStream performance is still acceptable, as seen in Figure 3(b) and Figure 3(a), both the throughput and goodput of the system degrade as the size of the overlay increases. Since all of the nodes in these experiments have good bandwidth and latency properties and we have shown that SplitStream performs well at a streaming rate of 1 Mbps, the goodput degradation can be attributed to the increase in overlay size.

We can see from Figure 3(c) and Figure 3(d) that both systems' ability to maintain consistently low amounts of late data is invariant of the overlay size. However, in general, Chainsaw is able to outperform SplitStream under large group sizes, maintaining a higher streaming rate and larger amount of good data in the system.

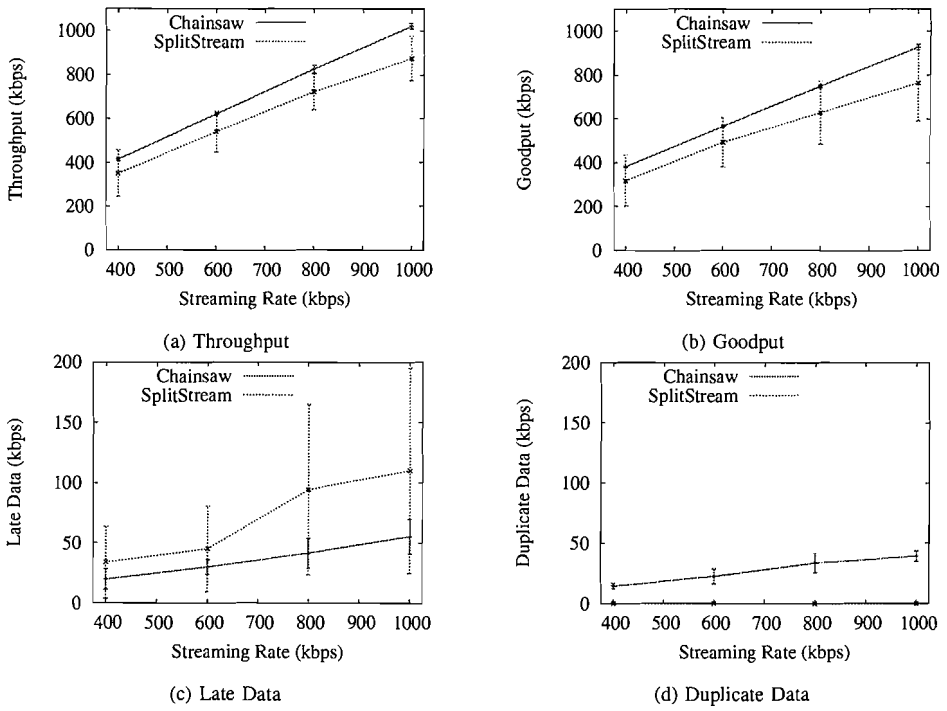


Fig. 2. Performance for different streaming rates using a configuration of 280 nodes with bandwidth greater than 1 Mbps

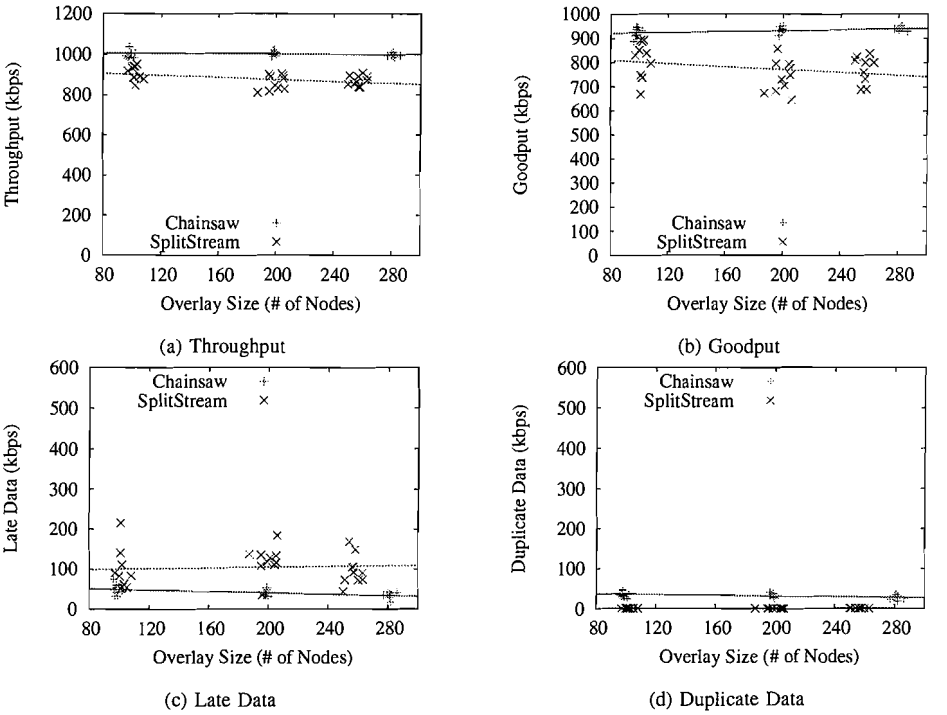
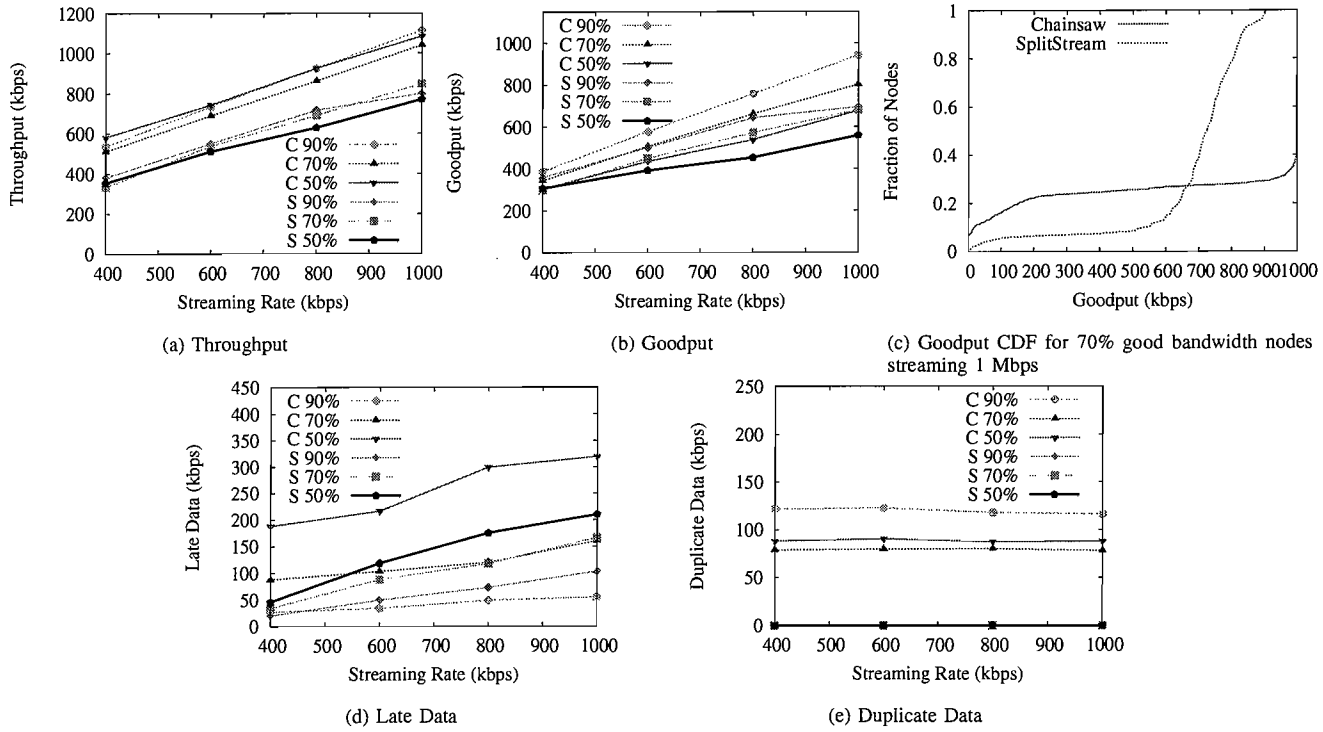


Fig. 3. Performance for different overlay sizes for a 1 Mbps streaming rate



(a) Throughput (b) Goodput (c) Goodput CDF for 70% good bandwidth nodes streaming 1 Mbps

(d) Late Data (e) Duplicate Data

Fig. 4. Performance for different percentages of nodes with bandwidth of 1 Mbps or higher, using a configuration of 112 nodes with heterogeneous bandwidth capabilities. In the figure, "C" denotes Chainsaw and "S" denotes SplitStream

D. Impact of Bandwidth Heterogeneity

Figure 4 demonstrates the impact of bandwidth heterogeneity on the performance of the multicast systems. Different percentages of full versus restricted bandwidth nodes were selected as overlay participants in our experiments. For example, in the 70% experiment, 30% of the nodes had varying bandwidth capabilities that were less than 1 Mbps, while the remaining nodes were more than capable of streaming the full system streaming rates. Nodes for each group were selected at random from nodes matching the bandwidth criteria. The source streamed data at rates between 400 kbps and 1 Mbps, and about 112 nodes participated in each of the experiments. To expedite the experiments we streamed for 10 minutes each.

As seen in Figure 4(a), the throughput varies little for each system, regardless of the percentage of bandwidth-constrained nodes. However, Figure 4(b) shows that the usefulness of the data decreases as streaming rates and percentage of bandwidth-constrained nodes increases. The reason for this can be seen in Figure 4(d). As the percentage of bandwidth-constrained nodes increases, the amount of late data considerably increases. This can be explained by the fact that in both systems, bandwidth-constrained peers become overwhelmed and quickly get behind on their duties to relay data to their peers. The amount of late data is significantly larger in Chainsaw than in SplitStream because if a packet is not received 1 second after the request, that same packet is requested again from another peer which can create another late packet. Thus, it would be worthwhile for each mesh node to keep track of an expected round-trip time between every peer and itself and intelligently schedule packets based on that value. This would also decrease the amount of duplicate data received. We have validated this by experimenting with timeouts of 2 seconds and 3 seconds, and found that the late and duplicate data indeed decreases.

Figure 4(c) characterizes how individual nodes perform in each system when 30% of the nodes are bandwidth-constrained (for the 1 Mbps streaming rate scenario). In SplitStream, very few nodes receive none of the stream and no nodes receive the entire stream. This is due to the fact that in a tree, all nodes are penalized if they have an ancestor that is bandwidth-constrained. In Chainsaw, about 70% of the nodes receive most of the stream (almost vertical line between 0.4 and 1 at 1 Mbps), while the rest receive very little of the stream (steep curve between 0 and 200 kbps). This demonstrates that Chainsaw mitigates the impact of bandwidth-constrained nodes on high bandwidth nodes. However, it also shows that Chainsaw penalizes low bandwidth nodes since they receive very little of the stream.

E. Impact of Latency Heterogeneity

Figure 5 demonstrates the impact of latency heterogeneity on the performance of Chainsaw and SplitStream when three-quarters (15 out of 20) of the nodes are in close proximity, and one-quarter have high latency in relation to the closely connected majority and the source. The 15 nodes in close proximity were located in North America with RTTs of less than 50 ms, and the rest of the nodes were selected at random from nodes in Europe with RTTs of greater than 150 ms to the source at Purdue University. The source streamed data at rates between 400 kbps and 1 Mbps for 10 minutes.

Interestingly, as seen in Figure 5(b), we find the SplitStream and Chainsaw goodput results are quite similar to each other in

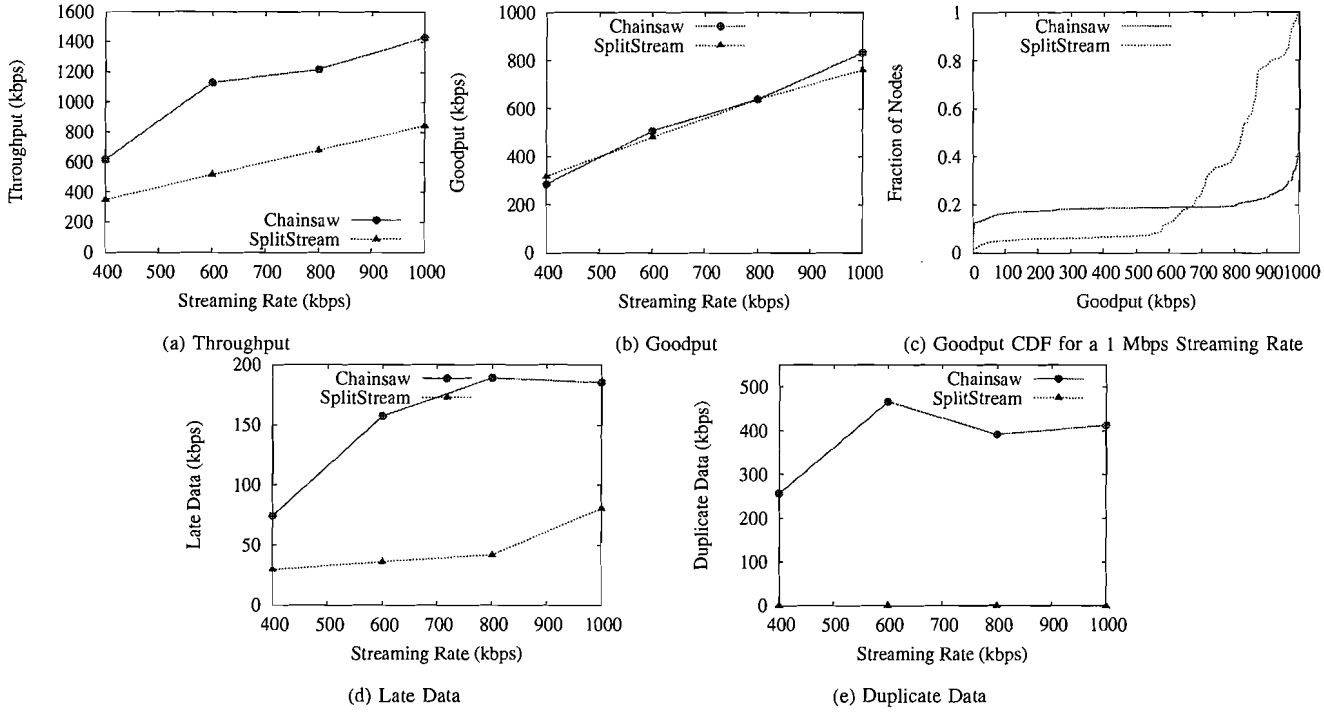


Fig. 5. Performance for a configuration of 20 nodes with heterogeneous latencies: 15 nodes are in close proximity to the source and each other, while the remaining 5 have longer latencies to these 15 nodes and the source

these experiments. However, even though the average system goodput is very similar, the individual node performance experienced when streaming 1 Mbps, presented in Figure 5(c), is quite dissimilar. Chainsaw exhibits two sets of nodes achieving two performance extremes, very low or very high throughput, while SplitStream nodes exhibit a much broader range of performance, with the majority of the nodes receiving between 600 kbps and 1 Mbps.

In contrast to the goodput, Figure 5(a) shows that the Chainsaw throughput is appreciably higher than that of SplitStream, due to a significant amount of late data (Figure 5(d)) and duplicate data (Figure 5(e)). This indicates that SplitStream is better able to push data to the nodes with longer RTTs within the deadlines, whereas the pull mechanism of Chainsaw causes several packet deadlines for the long latency nodes to be missed.

We repeated our experiments with a total of 70 and a total of 220 nodes, with 60% of the nodes being in North America and the rest in other continents. We found that as the overlay size increases, the average performance of Chainsaw increases and the average performance of SplitStream decreases, which is consistent with the results in Section IV-C. These results are omitted for space reasons, and because of their similarity to Figure 2.

F. Flash Crowds

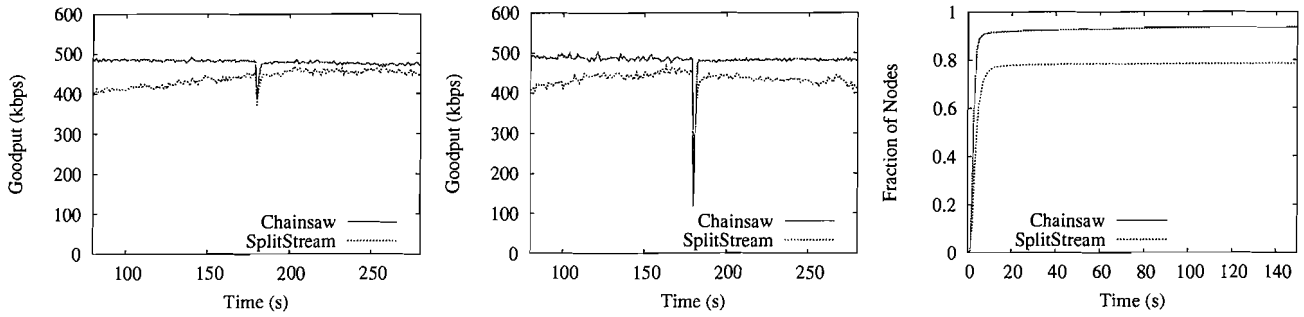
To determine the effect of flash crowds on the stability and performance of the multicast systems, we used about 280 nodes for each overlay and had a designated percentage of the nodes join midway through the experiment lifetime. The duration of the experiment was 6 minutes during which the source streamed data at 500 kbps. The system was allowed to stabilize before the flash crowd nodes join at 3 minutes after the experiment started in order to isolate the effect of the crowd.

Figure 6 depicts the effect that two exemplar percentages (flash crowds of 20% and 80% of the nodes) had on the two systems. From Figure 6(a) and Figure 6(b), we can see that both multicast systems quickly stabilize and return to performance levels similar to before the flash crowd, even when the majority of the nodes join after the experiment has begun. However, as seen from Figure 6(b), the performance of SplitStream begins to degrade with larger flash crowd sizes. We believe this is due to SplitStream attempting to find appropriate parents for nodes in the flash crowd, which can create a lengthy startup time for nodes.

We also examine the effects on the individual flash nodes after joining the network in order to determine what an individual user might experience. In Figure 6(c), we can see both systems are able to effectively integrate a majority of nodes into the dissemination structure and provide good performance to these nodes (within 90% of the streaming rate). In both cases, over 90% of the Chainsaw nodes and 75% of the SplitStream nodes achieve good performance. This difference in individual performance also helps explain why the average performance of SplitStream degrades with larger flash crowd sizes, since SplitStream has a larger percentage of nodes not receiving the desired bandwidth.

G. Churn

To evaluate the impact of churn on each overlay, we began with an overlay of 80 nodes. We then model node join behavior



(a) 20% Nodes in Flash Crowd (b) 80% Nodes in Flash Crowd (c) CDF of 80% Nodes in Flash Crowd

Fig. 6. Performance for different percentages of flash crowds using a configuration of 280 nodes with bandwidth greater than 1 Mbps

using a Poisson process and node stay time using a Pareto distribution. These choices were motivated by observations from real overlay multicast deployments [24] and Mbone sessions [40] and have been previously used by Bharambe *et al.* in [41]. For the Pareto distribution, we assume a minimum stay time of 90 seconds and an α of 1.42, which results in a mean stay time of 300 seconds. These parameters are consistent with distributions found in other live streaming applications on the Internet [42], [24]. We vary the mean of the Poisson process between 5 and 15, leading to group sizes varying from 150 to 280 nodes. For example, if the Poisson mean is set to 10, then on average, every 10 seconds there is a node that joins. Each experiment ran for 16 minutes and 40 seconds (1000 seconds) and the source streamed data at 500 kbps.

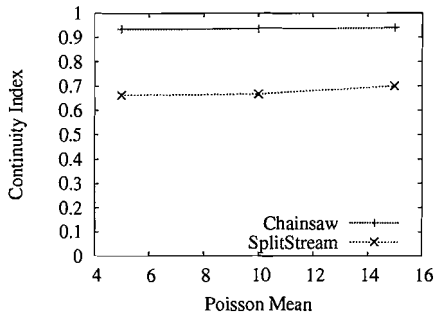


Fig. 7. Continuity Index using a configuration of an initial 80 nodes followed by other nodes joining based on a Poisson process

The results presented in Figure 7 indicate that Chainsaw performs better. First, it always has a much higher continuity index than SplitStream. This can be attributed to its receiving much more unique data than SplitStream. Second and more importantly, a higher join rate has a lower effect on it than on SplitStream. We can calculate from previous experiments that without churn, Chainsaw and SplitStream have a continuity index of .95 and .88, respectively. Hence, we can see that churn has a drastic effect on SplitStream. Since we have shown that SplitStream deals well with flash crowds, we attribute this to the time consuming process of children recognizing that their parent is gone and then reinserting each child and its subtree somewhere else. In contrast, Chainsaw nodes have many neighbors from whom to request packets and can also simultaneously request more neighbors from the bootstrap node.

V. Related Work

A number of studies have compared overlay multicast networks via simulations and on the Internet, including [14], [15], [16], [17]. These studies, however, focused on network-level metrics, such as the underlying overlay structure, relative delay penalty over unicast and IP multicast, and link stress (i.e., number of duplicate packets on each underlying Internet link). They did not consider application-level metrics for real-time streaming applications, as we do in this work.

Several other studies compared overlay networks for file-sharing applications [43], [44]. These studies focus on comparing unstructured networks similar in spirit to Gnutella, with structured overlay networks such as ones using distributed hash tables. Unlike these studies, we use a variety of application-specific metrics, since our focus is on streaming applications.

With the emergence of many, sometimes proprietary, commercial streaming systems, another focus of recent research has been understanding user behavior. Hei *et al.* [32] measured the performance of the PPLive [4] system, with the goal of quantifying user behavior and gaining insights into the protocol underlying PPLive. Deployments of open source systems have also been studied. Chu *et al.* [24] analyzed traces collected from a system based on ESM [12].

Perhaps closest to our work is the work in [18], which presents an interesting simulation comparison of a multi-tree scheme similar to SplitStream and the PRIME [19] overlay multicast proposal. Our goal, however, is to understand performance of existing streaming systems under a variety of real Internet conditions, including realistic latency and bandwidth heterogeneity. Further, we believe that Chainsaw is closer in spirit to a basic mesh, and hence more suited to our study.

VI. Conclusions

In this paper, we have compared the streaming performance of two representative P2P streaming systems, SplitStream and Chainsaw, via Internet experiments using PlanetLab. We can make several observations from our experimental results.

First, the mesh-based Chainsaw generally yields a higher goodput to the streaming application than the multi-tree based SplitStream. The difference between the two systems is small when streaming rates are low, or when the number of nodes in the system is small. However, Chainsaw scales better to higher streaming nodes or larger overlays.

Second, SplitStream was better able to cope with nodes that have higher latencies to the remaining nodes, while Chainsaw had a significant amount of late data and duplicate data in that case. The nodes with high latency suffered in performance with Chainsaw. In cases with bandwidth-limited nodes, Chainsaw performed better than SplitStream on the average, but bandwidth-limited nodes suffered, and Chainsaw again transmitted considerable late and duplicate data. Based on these results, we suggest that mesh-based systems use adaptive timeouts and intelligently schedule packets based on expected round-trip times. Third, as expected, Chainsaw was better able to deal with churn and with large flash crowds.

From our observations, mesh-based systems appear to be a better choice than multi-tree based systems for peer-to-peer streaming, especially for larger overlays and higher streaming rates. Mesh-based systems are clearly a better choice for nodes with high bandwidth capabilities and low round trip times, while multi-tree based systems currently cope better with stringent real time deadlines under heterogeneous conditions.

ACKNOWLEDGMENTS

This research is sponsored in part by NSF grants 0238294 and 0430271. We would like to thank the authors of Chainsaw and SplitStream for the code we used in our experiments.

REFERENCES

[1] http://www.joost.com.
[2] http://www.veoh.com.
[3] http://zattoo.com.
[4] http://www.pplive.com.
[5] http://www.ppstream.com.
[6] M. Castro, P. Druschel, A. M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," in *SOSP*, 2003.
[7] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *IPTPS*, 2005.
[8] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *IEEE ICNP*, 2006.
[9] X. Zhang, J. Liu, B. Li, and T.-S. P. Yum, "CoolStreaming/DONet: A data-driven overlay network for peer-to-peer live media streaming," in *IEEE INFOCOM*, 2005.
[10] D. Kostic, A. Rodriguez, J. Albrecht, and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *ACM SOSP*, 2003.
[11] C. L. Abad, W. Yurcik, and R. H. Campbell, "A survey and comparison of end-system overlay multicast solutions suitable for network-centric warfare," in *SPIE*, 2004.
[12] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proceedings of ACM SIGMETRICS*, 2000.
[13] B. Cohen, "Incentives build robustness in BitTorrent," in *Proceedings of P2P Economics*, 2003.
[14] S. Fahmy and M. Kwon, "Characterizing overlay multicast networks and their costs," *IEEE/ACM Transactions on Networking*, 2007.
[15] M. Castro, M. B. Jones, A. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlays," in *IEEE INFOCOM*, 2003.
[16] L. Lao, J.-H. Cui, M. Gerla, and D. Maggiorini, "A comparative study of multicast protocols: top, bottom, or in the middle?," in *IEEE INFOCOM*, 2005.
[17] S.-W. Tan, A. G. Waters, and J. S. Crawford, "A performance comparison of self-organising application layer multicast overlay construction techniques.," *Computer Communications*, 2006.
[18] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple trees: A comparative study of live P2P streaming approaches," in *IEEE INFOCOM*, 2007.
[19] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer receiver driven mesh-based streaming," in *IEEE INFOCOM*, 2007.
[20] B. Chun, D. Culler, T. Roscoe, A. Bavier, L. Peterson, M. Wawrzoniak, and M. Bowman, "Planetlab: an overlay testbed for broad-coverage services," *SIGCOMM Comput. Commun. Rev.*, 2003.
[21] S. Shi and J. Turner, "Routing in Overlay Multicast Networks," in *IEEE INFOCOM*, 2002.
[22] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. OToole Jr., "Overcast: Reliable multicasting with an overlay network," in *OSDI*, 2000.
[23] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *ACM SIGCOMM*, 2002.
[24] Y. Chu, A. Ganjam, T. S. E. Ng, S. G. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early experience with an internet broadcast system based on overlay multicast," in *USENIX Annual Technical Conference*, 2004.
[25] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *ACM/IEEE NOSSDAV*, 2002.
[26] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast," in *IEEE ICDCS*, 2007.
[27] B. Biskupski, R. Cunningham, J. Dowling, and R. Meier, "High-bandwidth mesh-based overlay multicast in heterogeneous environments," in *AAA-IDEA*, 2006.
[28] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat, "Using random subsets to build scalable network services," in *USENIX USITS*, 2003.
[29] V. Vishnumurthy and P. Francis, "On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks," 2006.
[30] A. I. T. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "SCRIBE: The design of a large-scale event notification infrastructure," in *NGC*, 2001.
[31] A. I. T. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *IFIP/ACM Middleware*, 2001.
[32] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale P2P IPTV system," *IEEE Transactions on Multimedia*, 2007.
[33] R. Chung, "EdgeStream Network Latency and Its Effect on Video Streaming," tech. rep., EdgeStream, 2004.
[34] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," in *ACM SIGCOMM*, 2004.
[35] N. Spring, L. Peterson, A. Bavier, and V. Pait, "Using PlanetLab for network research: Myths, realities, and best practices," *ACM SIGOPS*, 2006.
[36] A. Lo, G. Heijenk, and I. Niemegeers, "Evaluation of mpeg-4 video streaming over umts/wcdma dedicated channels," in *WICON*, 2005.
[37] L. Guo, E. Tan, S. Chen, Z. Xiao, O. Spatscheck, and X. Zhang, "Delving into internet streaming media delivery: a quality and resource utilization perspective," in *IMC*, 2006.
[38] "Buffer settings in windows media player, http://support.microsoft.com/kb/257535/."
[39] V. Pai and A. Mohr, "Improving robustness of peer-to-peer streaming with incentives," in *NetEcon*, 2006.
[40] K. C. Almeroth and M. H. Ammar, "Characterization of mbone session dynamics: Developing and applying a measurement tool," Tech. Rep. GIT-CC-95-22, Georgia Institute of Technology, 1995.
[41] A. Bharambe, S. Rao, V. Padmanabhan, S. Seshan, and H. Zhang, "The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols," in *IPTPS*, 2005.
[42] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *IMC*, 2004.
[43] M. Castro, M. Costa, and A. Rowstron, "Debunking some myths about structured and unstructured overlays," in *NSDI*, 2005.
[44] K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," *Communications Surveys & Tutorials, IEEE*, 2005.