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*—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 compare streaming overlay architectures in real Internet settings, considering 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. Overall, our study indicates that mesh-based systems are superior 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.**

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]–[16]. 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 [17]–[21] to meet the stringent demands of the commercial market, replacing network-layer (IP) multicast.

Internet P2P streaming has different requirements [22] 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 P2P streaming have emerged in recent years: tree-based and mesh-based architectures. A tree-based overlay constructs a tree rooted at the source which is broadcasting the stream. An example of a tree-based overlay is ESM [23]. In a single tree-based routing topology, leaf nodes do not forward data, leading to an imbalance in the load on the peers. Recent research has introduced multi-tree overlays which distribute bandwidth costs across peers by disseminating the data on multiple dissimilar trees. Examples of multi-tree overlays are Chunkyspread [19] and SplitStream [17].

A mesh-based overlay enables data flow in a less structured manner, by requiring peers to exchange data with a subset of the nodes in the network. Examples of mesh-based multicast overlays are Chainsaw [18] and CoolStreaming/DONet [20]. Several popular IPTV systems, such as PPLive [2] and PPStream [3], extend ideas from the mesh-based BitTorrent [24] 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 design performs 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 [25]–[28], but none focused on streaming applications. Two exceptions are the work in [18] which presents a comparison of only the goodput of two systems – a mesh and a multi-tree system – under only one streaming rate, and the work in [29] which conducts a simulation comparison of a multi-tree system similar to SplitStream and PRIME [30] – 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 [31]. We select the mesh-based Chainsaw [18] and the tree-based SplitStream [17] systems because their core designs are based on a basic bidirectional mesh or a basic multi-tree topology. Additionally, they are the latest systems for which a complete implementation is publicly available. We compare their scalability with increasing application-prescribed streaming rates and number of users, amount of unusable data, impact of bandwidth and latency heterogeneity, system recovery when confronted with flash crowds, and performance under churn. To the best of our knowledge, this is the first comprehensive 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. We consider 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. We summarize our findings as follows:

- The mesh-based Chainsaw generally yields a higher goodput to the streaming application than the multi-tree based SplitStream. Although the difference between the two systems is small when streaming rates are low, or when the number of nodes in the system is small, Chainsaw scales better to higher streaming rates and larger overlays. Surprisingly, we were not able to find any saturation point for streaming rates varying from 400 kbps to 1 Mbps.

- When deployed in networks with heterogeneous latencies, SplitStream is better able to cope with nodes that have higher latencies to the remaining nodes, while Chainsaw has a significant amount of late data and duplicate data in that case. Nodes with limited access bandwidth also suffer more with Chainsaw than with SplitStream. Based on these results, we suggest that mesh-based systems use adaptive timeouts.
- Chainsaw is better able to deal with churn and with large flash crowds since its overlay management is less complex. SplitStream is more sensitive to flash crowds and particularly churn, as the cost of constructing the trees is higher.

II. Taxonomy of Overlay Multicast Approaches

The earliest multicast overlays used a single-tree topology, and did not specifically address real time streaming requirements [23], [32]–[34]. For example, Overcast [33] was designed for reliable communication, such as file distribution. Later, some of the overlay multicast systems were extended for the application of Internet streaming; for example, ESM [23] was extended and deployed for streaming as discussed in [35]. Multi-tree systems such as CoopNet [36], SplitStream [17], and Chunkyspread [19] were later proposed to distribute bandwidth costs across overlay participants.

Mesh-based systems, e.g., CoolStreaming/DONet [20], and Chainsaw [18], were proposed to address the inherent lack of resilience of tree-based structures. Hybrid systems such as Bullet [21] and mTreebone [37] 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 [38] and PRIME [30] 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 classifies several popular overlay multicast approaches according to the mechanisms they employ. In the table, "peer discovery" refers to how each node finds new neighbors after it has joined the overlay. RanSub [39] and SwapLinks [40] are distributed algorithms that find nodes to peer with. Based on this classification, we have selected Chainsaw [18] and SplitStream [17] for our experiments, because their core design reflects a basic bidirectional mesh or multi-tree topology.

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			

III. Comparing P2P Streaming Approaches

We next discuss the two systems we have selected for our study, and present the criteria by which we compare them.

A. Chainsaw

Chainsaw [18] 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.

A node notifies its neighbors whenever he receives a new packet. 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. Nodes discard packets after some 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. 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 [17] is a single-source, multiple-receiver, multi-tree overlay utilizing a push-based approach in which the source disseminates data over several interior-node 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 [41], an application level multicast infrastructure that is itself built on top of the Pastry Distributed Hash Table (DHT) [42].

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. A node immediately forwards 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

P2P systems are expected to scale well with the number of participants and take advantage of the resources contributed by each participant. Internet *streaming applications* have also 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 [43], with 1 Mbps delivering "TV quality" audio and video [44], 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 meet these requirements, streaming overlays often duplicate data in the network, resulting in traffic which may not be useful from the application perspective.

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. We note that although several optimizations were recently proposed to better use available bandwidth and not penalize nodes with low bandwidth [19], [30], [45], we focus on studying which overlay strategy is inherently better able to handle bandwidth heterogeneity. We defer studying which optimization technique works best for an overlay architecture (mesh-based or multi-tree) to future work.

(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 [46]. 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

A. Experimental Methodology

To study the two systems under real-world conditions, we conducted our experiments on PlanetLab [31]. PlanetLab provides a research platform for large scale distributed experimentation of peer-to-peer systems over the Internet [47]. In order to mitigate the possible limitations of using a testbed, such as those addressed in [47], we ran experiments at different times of the day and different days of the week and found there is little variability in the systems with respect to the time or day the experiments were performed. Further, we randomly selected nodes for different experiments (subject to certain constraints as discussed later in this section) to validate the statistical significance of results. Nodes were chosen to span multiple operational and administrative domains. Each experiment was repeated ten times and we show confidence intervals and statically validate the significance of our results.

For both systems, the source sends streaming bit rates of 400 kbps to 1 Mbps, which are representative of the streaming rates currently used in many video streaming applications [43]. The source was located on a host at Purdue University. We configured the source to wait for 30 seconds before starting to send data to allow for the application to stabilize.

Both systems use the TCP transport protocol. We consider that a packet must arrive within 5 seconds to be considered useful, according to the buffer times used in [48]–[50]. Our experiments with 10 and 15 second thresholds revealed that both systems perform only marginally better, so we omit these results from the paper. We used a maximum of 280 nodes in our experiments because that is the largest number of PlanetLab nodes that were responsive and satisfied our bandwidth and latency requirements discussed below. Although this may not seem like a very large number, it still reveals interesting real-life scenarios, while allowing us to control the characteristics of the nodes [47].

In Chainsaw, 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 used a default data "chunk" size of 2500 bytes for Chainsaw. We use the terms "chunk" and "packet" interchangeably throughout the paper. These parameters are identical to those used in previous Chainsaw deployments [18], [51].

For SplitStream, the source sends one chunk of the data stream down each tree every second, with each node forwarding that chunk onto each of its children as soon as possible. Each SplitStream node is configured to accept between 16 and 48 children, depending on the available bandwidth of the node. SplitStream uses 16 trees by default, and we show results with 16 trees in this paper. We have also experimented with 8 trees and 32 trees, validating that 16 trees and 32 trees outperform 8 trees. The variability in the results significantly decreases as the number of trees increases at the expense of increased management overhead. Our results confirm that SplitStream with 16 trees presents a good tradeoff. All parameters mirror those used in previous SplitStream deployments [17].

We compare the systems based on the following metrics:

- **Goodput** is the average rate of data that was received before the deadline (5 sec.), 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 Zhang *et al.* [20], 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 $\frac{Goodput}{StreamingRate}$. Ideally, a system should have a continuity index of 1.

B. Scalability with Streaming Rates

We first compare how well each overlay scales with increasing streaming rates. To decouple the scalability with the streaming rate from the impact of heterogeneity, we use nodes that have high access bandwidth (greater than 1 Mbps) and low latency (with average Round Trip Times (RTTs) of 100 ms to the source). This amounted to a deployment of 280 nodes on PlanetLab that were responsive and met the stated bandwidth and latency requirements. We vary the streaming rate from 400 kbps to 1 Mbps. Figure 1 shows the mean system performance with 90% confidence intervals.

Figure 1(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 of Chainsaw is close to the ideal. In contrast, the SplitStream throughput is considerably less than the ideal, especially as the streaming rate increases. As seen in Fig. 1(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. The SplitStream performance is more variable across multiple experiments, demonstrating SplitStream is more sensitive to network conditions. We also note that at lower streaming rates, there is a small overlap between the error bars. We further investigate if the means of the results for the two systems are different. We formulate the null hypothesis $H_0 : \mu_{Chainsaw} = \mu_{SplitStream}$, which states Chainsaw throughput (or goodput) and SplitStream throughput (or goodput) have the same mean and distribution. Using a two-sample t-test with pooled variance [52], we disproved H_0, finding with high probability that the error results come from distributions with different means.

The explanation behind the lower goodput of SplitStream is depicted in Fig. 1(c). SplitStream receives a non-negligible amount of late data (data received after the 5 sec. deadline), which is higher than the late data received by Chainsaw. As both systems use TCP, we attribute the difference to the fact that Chainsaw is a pull-based system, where each peer decides what pieces of information it needs, while SplitStream uses a push-based approach in which nodes push data to their children on different trees at different times. This, combined with SplitStream's lack of any *mechanism for dropping late data*, results in unnecessary bandwidth consumption. The unnecessary overhead is also amplified by the fact that SplitStream does not change its trees unless nodes leave the overlay.

We also note an increasing variability in the late data for SplitStream. Further investigation reveals that the average depth of SplitStream trees is not highly variable with a mean depth of 3.5 and a standard deviation of 1. Therefore, this variability stems from network conditions delaying data that continues to be late as it propagates down the tree. In contrast, Chainsaw is able to request missing data from neighbors, and obtain the missing data in time. As in the case for throughput and goodput, we validate that the means corresponding to the experiments for the two systems are statistically different by using a two-sample t-test with pooled variance.

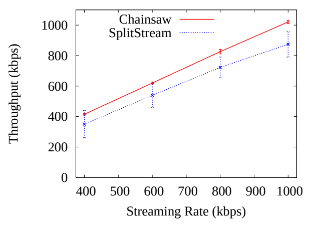
Figure 1(d) shows the duplicate data for both overlays. SplitStream, being tree-based, receives a negligible amount of duplicate data, whereas Chainsaw suffers from a growing amount of duplicate data as streaming rates increase. SplitStream received negligible amounts of duplicate data in all the experiments 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 neither system has an inherent streaming rate below 1 Mbps where it cannot send any more data. In addition, we found that SplitStream is more sensitive to network conditions, generating a higher amount of late data.

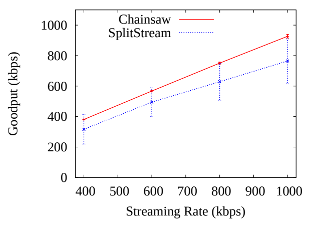
C. Scalability with Overlay Size

Figure 2 shows the impact of the size of the multicast group when using a streaming rate of 1 Mbps for 20 minutes, varying the overlay size from 80 to 280 nodes. We also repeated the set of experiments for a streaming rate of 500 kbps and the results were similar (with SplitStream and Chainsaw being closer in performance). We omit these results due to space limitations.

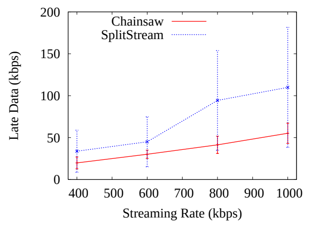
As the overlay size increases, Fig. 2(b) shows that the goodput of Chainsaw slightly increases, without a corresponding increase in throughput (Fig. 2(a)). This demonstrates



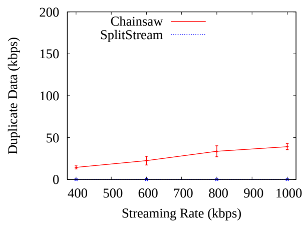
(a) Throughput



(b) Goodput

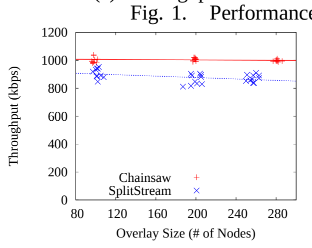


(c) Late Data

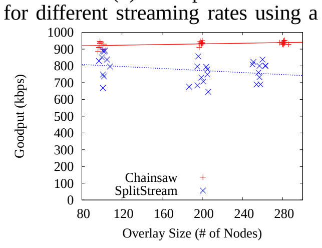


(d) Duplicate Data

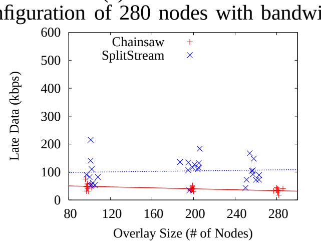
Fig. 1. Performance for different streaming rates using a configuration of 280 nodes with bandwidth greater than 1 Mbps



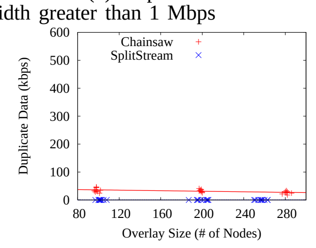
(a) Throughput



(b) Goodput

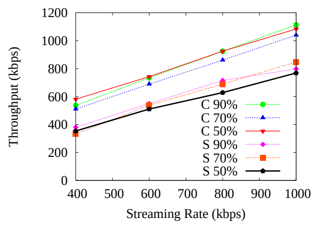


(c) Late Data

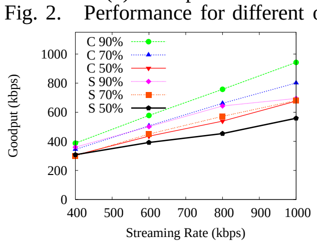


(d) Duplicate Data

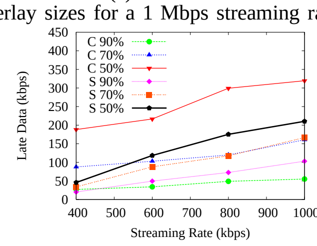
Fig. 2. Performance for different overlay sizes for a 1 Mbps streaming rate



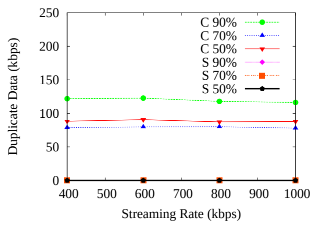
(a) Throughput



(b) Goodput



(c) Late Data



(d) Duplicate Data

Fig. 3. 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

that Chainsaw scales with the number of nodes and is able to effectively use the available resources without increasing the late or duplicate data. This is not the case with SplitStream. Although SplitStream performance is still acceptable, (Fig. 2(a) and 2(b)), the throughput and goodput of the system degrade as the number of nodes 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 (Fig. 1(b)), we attribute the goodput degradation to the increase in overlay size.

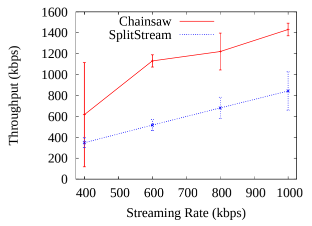
We can see from Fig. 2(c) and 2(d) that both systems' ability to maintain consistently low amounts of late data is invariant of the overlay size. However, in general, Chainsaw is able to outperform SplitStream under large group sizes, maintaining a higher streaming rate and larger amount of useful data.

D. Impact of Bandwidth Heterogeneity

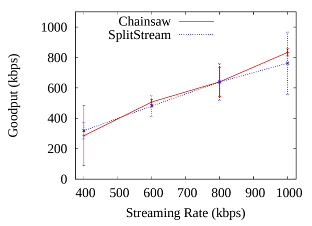
In order to study the effect of bandwidth heterogeneity on the performance of the multicast systems we first group nodes in two classes, nodes with low bandwidth (under 1 Mbps) and high bandwidth (over 1 Mbps). In Fig. 3, we present the results for deployments with different ratios of low-bandwidth and high-bandwidth nodes. For example, the line denoted "C 70%" (or "S 70%") represents a Chainsaw (or SplitStream) experiment where 70% of nodes were high-bandwidth nodes and 30% of the nodes were low-bandwidth nodes. Nodes for each group were selected at random from nodes matching the bandwidth criteria, with approximately 112 nodes participating in each experiment. To expedite the experiments, the source streamed data at rates between 400 kbps and 1 Mbps for 10 minutes. We omit confidence intervals from these graphs as they become difficult to read otherwise.

As seen in Fig. 3(a), the throughput varies little for each system, regardless of the percentage of bandwidth-constrained nodes. However, Fig. 3(b) shows that the usefulness of the data decreases as streaming rates and percentage of bandwidth-constrained nodes increases. As seen in Fig. 3(c), as the percentage of bandwidth-constrained nodes increases, the amount of late data increases considerably. This can be explained by the fact that in both systems, bandwidth-constrained peers become overwhelmed and fall behind on their duties to relay data to their peers. The amount of late data is significantly larger in Chainsaw than SplitStream because if a packet is not received 1 second after the request, that same packet is requested from another peer, which may yield yet 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 hypothesis by experimenting with timeouts of 2 seconds and 3 seconds, and found that the late and duplicate data indeed decrease.

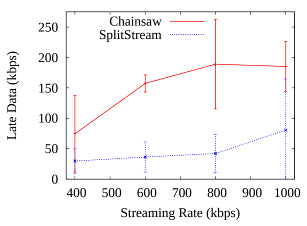
Figure 5 characterizes how individual nodes perform in each system when 30% of the nodes are bandwidth-constrained (for the 1 Mbps streaming rate scenario). In SplitStream, very few nodes receive none of the stream and no nodes receive the entire stream. This is due to the fact that in a tree, all nodes are penalized if they have an ancestor that is bandwidth-



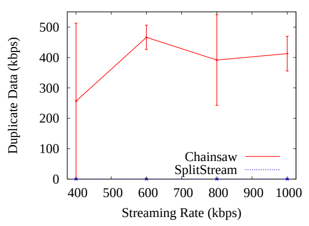
(a) Throughput



(b) Goodput



(c) Late Data



(d) Duplicate Data

Fig. 4. 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

constrained. In Chainsaw, approximately 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, but penalizes low bandwidth nodes since they receive little data.

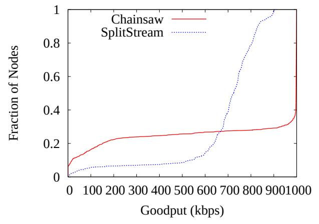


Fig. 5. Goodput CDF where the source streams 1 Mbps, using a configuration of 112 nodes with heterogeneous bandwidth capabilities. 70% of the nodes have bandwidth capabilities of 1 Mbps or higher.

E. Impact of Latency Heterogeneity

In order to study the impact of latency heterogeneity, we group nodes in two classes, close-proximity nodes (RTTs less than 50 ms) and low-proximity nodes (RTTs greater than 150 ms), with respect to the source located at Purdue University. We then consider overlay deployments in which we vary the fraction of close-proximity and low-proximity nodes. Figure 4 demonstrates the impact of latency heterogeneity on the performance of Chainsaw and SplitStream when three-quarters (15 of 20) of the nodes are close-proximity nodes and one-quarter have high latency in relation to the closely connected majority and the source. The source streamed rates between 400 kbps and 1 Mbps for 10 minutes.

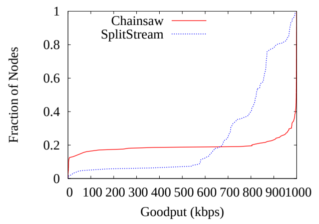


Fig. 6. Goodput CDF for a 1 Mbps streaming rate for a configuration of 20 nodes with heterogeneous latencies: 15 nodes are in close proximity to the source and each other, while the remaining 5 have longer latencies to these 15 nodes and the source

Interestingly, as seen in Fig. 4(b), we find the SplitStream and Chainsaw goodput results are quite similar to each other. However, even though the goodput is very similar, the individual node performance experienced when streaming 1 Mbps, presented in Fig. 6, 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.

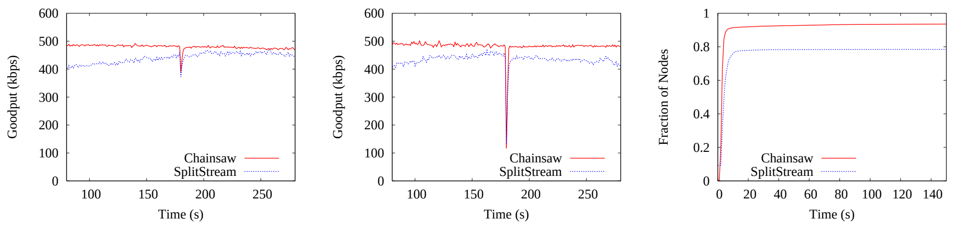
In contrast to the goodput, Fig. 4(a) shows that Chainsaw throughput is appreciably higher than that of SplitStream, due to a significant amount of late data (Fig. 4(c)) and duplicate data (Fig. 4(d)). This indicates that SplitStream is better able to push data to the nodes with longer RTTs within the deadlines, whereas the pull mechanism of Chainsaw causes packet deadlines for the high latency nodes to be missed. This can be attributed to the fact that Chainsaw neighbor sets are small and do not have highly disjoint data in these cases, which also explains the high variability in Chainsaw late and duplicate data. Further experiments where Chainsaw was configured with a higher timeout value (3 sec.) to re-request a packet showed increased variability in system performance, while throughput and goodput decreased. This confirms the high sensitivity of Chainsaw to the timeout to request packets, particularly in latency heterogeneous networks.

We repeated our experiments with overlays of 70 and 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, as they are similar to Fig. 1.

F. Flash Crowds

To determine the effect of flash crowds on the stability and performance of the systems, we used overlays of 280 nodes and designated a percentage of the nodes to 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 for 3 minutes before the flash crowd nodes joined to isolate the effect of the crowd.

Figure 7 depicts the effect that two exemplar percentages (flash crowds of 20% and 80% of the nodes) had on the two systems. From Fig. 7(a) and 7(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



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

Fig. 7. Performance for different percentages of flash crowds using a configuration of 280 nodes with bandwidth greater than 1 Mbps

seen from Fig. 7(b), the performance of SplitStream begins to degrade with larger flash crowd sizes. This can be attributed to SplitStream attempting to find 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 a user might experience. In Fig. 7(c), we can see both systems are able to effectively integrate a majority of nodes into the dissemination structure In both cases, over 90% of the Chainsaw nodes and 75% of the SplitStream nodes achieve good performance within 90% of the streaming rate. 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

We began with an overlay of 80 nodes and model node join behavior using a Poisson process and node stay time using a Pareto distribution. These choices were motivated by observations from real overlay multicast deployments [35] and Mbone sessions [53] and have been previously used by Bharambe *et al.* [54]. For the Pareto distribution, we assume a minimum stay time of 90 seconds and an α of 1.42, resulting in a mean stay time of 300 seconds. These parameters are consistent with distributions found in other live streaming applications on the Internet [35], [55]. We vary the mean inter-arrival time (λ) between 5 and 15 seconds; for example, if the mean inter-arrival time is set to 10, then on average, every 10 seconds a node joins. This leads to group sizes that vary between 150 and 280 nodes. We use ungraceful departures, i.e., nodes leave without informing their children, neighbors, or the source about their intention to leave. Each experiment ran for 1000 seconds and the source streamed data at 500 kbps.

TABLE II
CONTINUITY INDEX FOR BOTH SYSTEMS UNDER CHURN

System	$\lambda = 5$	$\lambda = 10$	$\lambda = 15$
Chainsaw	.94	.93	.94
SplitStream	.66	.67	.70

As can be seen in Table II, under the highest join rate, Chainsaw receives much more unique data than SplitStream, achieving continuity indexes of .94 and .66 respectively . Second and more importantly, a higher join rate has a lower effect on Chainsaw than SplitStream. We find the continuity index without churn for Chainsaw and SplitStream to be .95 and .88, respectively. The presence of churn decreases the continuity index of SplitStream by .22 while the continuity index of Chainsaw suffer almost no change. We conclude 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 children need to detect the failure of the their parent and then reinsert 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

Several studies [25]–[28] have compared overlay multicast networks via simulations and on the Internet. These studies focused on network-level metrics, such as the underlying overlay structure, relative delay penalty over unicast and IP multicast, and link stress. They did not consider application-level metrics for streaming applications, unlike our work.

Other studies compared overlay networks for file-sharing applications [56], [57]. 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. There have been studies that measured [43] the performance of the PPLive [2] system, with the goal of quantifying user behavior and gaining insights into the protocol underlying PPLive; analyzed [58] traces for UUSee to characterize the mesh topology and understand behavior with flash crowds; or quantified [59] the impact of caching techniques to reduce duplicate packets in bandwidth-constrained wireless LANs. Deployments of open source systems have also been studied [35].

Closest to our work is the work in [29], which presents a simulation comparison of a multi-tree scheme similar to SplitStream and the PRIME [30] overlay multicast. Our goal, however, is to understand performance of existing and publicly available streaming systems under a variety of real Internet conditions, including latency and bandwidth heterogeneity.

VI. Conclusions

We have compared the performance of two representative P2P streaming systems, SplitStream and Chainsaw, via Internet experiments using PlanetLab. We summarize our findings:

- The mesh-based Chainsaw generally yielded a higher goodput to the 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 rates and larger overlays. Performance of SplitStream may be improved by making the *trees adapt to network conditions* to decrease the amount of late data.

- SplitStream coped better with nodes that had higher latencies, while Chainsaw had a significant amount of late and duplicate data. In cases with bandwidth-limited nodes, Chainsaw performed better than SplitStream on the average, but bandwidth-limited nodes suffered. We suggest that mesh-based systems *use adaptive timeouts and intelligently schedule packets based on expected round-trip times*.
- As expected, Chainsaw dealt better with churn and with large flash crowds as it requires less complex management of the overlay and can easily accommodate these events. SplitStream was more sensitive to flash crowds and particularly churn, since the cost of changing the trees was higher.

REFERENCES

[1] SOPCast. http://www.sopcast.org/.
[2] PPLive. http://www.pplive.com.
[3] PPStream. http://www.ppstream.com.
[4] TVAnts. http://www.tvants-ppstream.com.
[5] QQLive. http://tv.qq.com.
[6] Feidian. http://www.feidian.com.
[7] Mysee. http://www.mysee.com.
[8] Pdbox. http://www.pdbox.co.kr.
[9] PPMate. http://www.ppmate.com.
[10] UUSee. http://www.uusee.com.
[11] VGO. http://vgo.21cn.com.
[12] CTV. http://www.tvoon.de/ctv.
[13] StreamerOne. http://www.streamerone.com.
[14] TVUnetworks. http://www.tvunetworks.com.
[15] Joost. http://www.joost.com.
[16] Zattoo. http://zattoo.com.
[17] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh, "Splitstream: High-bandwidth multicast in cooperative environments," in *SOSP*, 2003.
[18] V. Pai, K. Kumar, K. Tamilmani, V. Sambamurthy, and A. Mohr, "Chainsaw: Eliminating trees from overlay multicast," in *IPTPS*, 2005.
[19] V. Venkataraman, K. Yoshida, and P. Francis, "Chunkyspread: Heterogeneous unstructured tree-based peer-to-peer multicast," in *ICNP*, 2006.
[20] 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 *INFOCOM*, 2005.
[21] D. Kostic, A. Rodriguez, J. Albrecht, , and A. Vahdat, "Bullet: High bandwidth data dissemination using an overlay mesh," in *SOSP*, 2003.
[22] C. Abad, W. Yurcik, and R. Campbell, "A survey and comparison of end-system overlay multicast solutions suitable for network-centric warfare," in *Proc. of SPIE*, 2004.
[23] Y. Chu, S. G. Rao, and H. Zhang, "A case for end system multicast," in *Proc. of ACM SIGMETRICS*, 2000.
[24] B. Cohen, "Incentives build robustness in BitTorrent," in *Proc. of P2P Economics*, 2003.
[25] S. Fahmy and M. Kwon, "Characterizing overlay multicast networks and their costs," *IEEE/ACM Trans. Netw.*, vol. 15, pp. 373–386, 2007.
[26] M. Castro, M. Jones, A.-M. Kermarrec, A. Rowstron, M. Theimer, H. Wang, and A. Wolman, "An evaluation of scalable application-level multicast built using peer-to-peer overlays," in *INFOCOM*, 2003.
[27] L. Lao, J. Cui, M. Gerla, and D. Maggiorini, "A comparative study of multicast protocols: top, bottom, or in the middle?," in *INFOCOM*, 2005.
[28] S.-W. Tan, A. G. Waters, and J. S. Crawford, "A performance comparison of self-organising application layer multicast overlay construction techniques.," *Computer Communications*, vol. 29, pp. 2322–2347, 2006.
[29] N. Magharei, R. Rejaie, and Y. Guo, "Mesh or multiple trees: A comparative study of live P2P streaming approaches," in *INFOCOM*, 2007.
[30] N. Magharei and R. Rejaie, "PRIME: Peer-to-peer receiver driven mesh-based streaming," in *INFOCOM*, 2007.
[31] PlanetLab. http://www.planetlab.org.
[32] S. Shi and J. Turner, "Routing in Overlay Multicast Networks," in *INFOCOM*, 2002.
[33] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O. Jr., "Overcast: Reliable multicasting with an overlay network," in *OSDI*, 2000.
[34] S. Banerjee, B. Bhattacharjee, and C. Kommareddy, "Scalable application layer multicast," in *Proc. of ACM SIGCOMM*, 2002.
[35] Y.-H. Chu, A. Ganjam, T. S. E. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang, "Early experience with an internet broadcast system based on overlay multicast," in *Proc. of USENIX*, 2004.
[36] V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, "Distributing streaming media content using cooperative networking," in *Proc. of ACM/IEEE NOSSDAV*, 2002.
[37] F. Wang, Y. Xiong, and J. Liu, "mTreebone: A Hybrid Tree/Mesh Overlay for Application-Layer Live Video Multicast," in *ICDCS*, 2007.
[38] B. Biskupski, R. Cunningham, J. Dowling, and R. Meier, "High-bandwidth mesh-based overlay multicast in heterogeneous environments," in *Proc. of AAA-IDEA*, 2006.
[39] D. Kostic, A. Rodriguez, J. Albrecht, A. Bhirud, and A. Vahdat, "Using random subsets to build scalable network services," in *USITS*, 2003.
[40] V. Vishnumurthy and P. Francis, "On Heterogeneous Overlay Construction and Random Node Selection in Unstructured P2P Networks," in *INFOCOM*, 2006.
[41] A. Rowstron, A.-M. Kermarrec, M. Castro, and P. Druschel, "SCRIBE: The design of a large-scale event notification infrastructure," in *Proc. of NGC*, 2001.
[42] A. Rowstron and P. Druschel, "Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems," in *Proc. of IFIP/ACM Middleware*, 2001.
[43] X. Hei, C. Liang, J. Liang, Y. Liu, and K. W. Ross, "A measurement study of a large-scale p2p iptv system," *IEEE Trans. on Multimedia*, vol. 9, pp. 1672 – 1687, 2007.
[44] R. Chung, "EdgeStream Network Latency and Its Effect on Video Streaming," tech. rep., EdgeStream, 2004.
[45] M. Bishop, S. Rao, and K. Sripanidkulchai, "Considering priority in overlay multicast protocols under heterogeneous environments.," in *INFOCOM*, 2006.
[46] K. Sripanidkulchai, A. Ganjam, B. Maggs, and H. Zhang, "The feasibility of supporting large-scale live streaming applications with dynamic application end-points," *SIGCOMM Comput. Commun. Rev.*, vol. 34, pp. 107–120, 2004.
[47] N. Spring, L. Peterson, A. Bavier, and V. Pait, "Using PlanetLab for network research: Myths, realities, and best practices," *ACM SIGOPS Operating Systems Review*, vol. 40, pp. 17–24, 2006.
[48] A. Lo, G. Heijenk, and I. Niemegeers, "Evaluation of mpeg-4 video streaming over umts/wcdma dedicated channels," in *WICON*, 2005.
[49] 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 *Proc. of IMC*, 2006.
[50] "Buffer settings in windows media player." http://support.microsoft.com/kb/257535/.
[51] V. Pai and A. Mohr, "Improving robustness of peer-to-peer streaming with incentives," in *Proc. of NetEcon*, 2006.
[52] D. Moore and G. McCabe, *Introduction to the practice of statistics*. WH Freeman and Company, 2003.
[53] K. Almeroth and M. Ammar, "Characterization of mbone session dynamics: Developing and applying a measurement tool," Tech. Rep. GIT-CC-95-22, Georgia Institute of Technology, 1995.
[54] A. Bharambey, S. Rao, V. Padmanabhan, S. Seshan, and H. Zhang, "The impact of heterogeneous bandwidth constraints on DHT-based multicast protocols," in *Proc. of IPTPS*, 2005.
[55] K. Sripanidkulchai, B. Maggs, and H. Zhang, "An analysis of live streaming workloads on the internet," in *Proc. of IMC*, 2004.
[56] M. Castro, M. Costa, and A. Rowstron, "Debunking some myths about structured and unstructured overlays," in *Proc. of NSDI*, 2005.
[57] E. 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*, vol. 7, pp. 72–93, 2005.
[58] C. Wu, B. Li, and S. Zhao, "Magellan: Charting large-scale peer-to-peer live streaming topologies," in *ICDCS*, 2007.
[59] E. Tan, L. Guo, S. Chen, and X. Zhang, "SCAP: Smart caching in wireless access points to improve P2P streaming," in *ICDCS*, 2007.