Bandwidth Adaptation in Streaming Overlays

Simon Schubert*, Frank Uyeda†, Nedeljko Vasić*, Naveen Cherukuri‡, and Dejan Kostić*
*EPFL, Lausanne, Switzerland, first.last@epfl.ch
†UCSD, San Diego, USA, fuyeda@ucsd.edu (work done during an Internship at Microsoft Research)
‡UIUC, Urbana-Champaign, USA, cheruku3@illinois.edu (work done during an Internship at EPFL)

***Abstract*— A major challenge for real-time streaming overlays is to distribute high bit-rate streams with uninterrupted playback. Hosts usually have sufficient inbound bandwidth to support streaming, but due to the prevalence of asymmetric links in broadband networks, the bottleneck is the aggregate, overlay-wide outbound bandwidth. If this bandwidth is less than what is required to forward the stream to the overlay members, then a large number of users potentially experience poor playback. We argue that for successful streaming in bandwidth constrained situations overlays need to be able to adapt to the aggregate available bandwidth. We present four bandwidth adaptation policies for tree-based streaming overlays and evaluate their efficiency using a large-scale emulation testbed with realistic broadband link characteristics.**

I. Introduction

It is becoming increasingly popular to use host-based overlays for real-time media streaming. A major challenge for these streaming overlays is to manage the available bandwidth resources in the overlay. With significant numbers of hosts behind asymmetric links, it is often the case that the aggregate outbound bandwidth available is an order of magnitude smaller than the aggregate inbound bandwidth. If there is insufficient outbound bandwidth then the overlay cannot successfully forward the stream to all members of the overlay, resulting in poor playback performance for many hosts. Therefore, the bottleneck for streaming overlays is the aggregate outbound bandwidth, which has been observed in deployed systems [10], [18]. This problem is further compounded as content providers want to increase streaming bit rates to enable distribution of high definition video, given that most users have sufficient inbound bandwidth to view it.

The majority of deployed streaming overlays, such as PPLive [23], either assume that there is sufficient aggregate outbound bandwidth available to the overlay, or rely on the goodwill of university-based hosts or dedicated infrastructure (such as PlanetLab or a CDN) to provide extra forwarding capacity to ensure continuous playback for all overlay members. As commercial operators begin to use streaming overlays, the gross exploitation of altruistic members (such as those running at universities) is becoming increasingly unacceptable (at least to those who pay for the bandwidth). For example, many universities have already started blocking certain types of traffic associated with overlay services, such as Skype relaying and content distribution systems. We believe that streaming overlays need to be able to dynamically adapt to the available bandwidth.

In this paper, we present four specific overlay bandwidth adaptation policies, for use in tree-based streaming overlays:

Node contribution adaptation enables each node to dynamically adapt the amount of bandwidth it contributes to the overlay, such that individual nodes can locally alter their contribution to maintain a particular bandwidth contribution skew across all nodes. For example, this can be used to implement fairness policies, but can also be used to control when and how much capacity a CDN should contribute.

Dynamic stream rate selection enables the source of a media stream to select an encoding bit rate that is highest at which all members of the overlay can be successfully supported.

Admission control allows each member of the overlay to determine if there is sufficient capacity to support a joining node. This policy is important during periods of high churn in single trees and in general in multi-tree overlays, where a node needs to be joined in all trees or rejected completely.

Per-organization bandwidth limitation enables administrators to limit the aggregate outbound bandwidth of their organization used by the overlay. At the same time, this policy enables locality-aware construction of the overlay tree which prefers intra-organization connections and avoids inter-organization traffic. This is a viable alternative to blocking overlay traffic because it allows members to participate in the overlay stream, without involuntarily turning their organization into an altruistic bandwidth provider.

These policies make different resource usage tradeoffs, e.g., they let the content provider decide whether it is more important to provide highest-possible streaming rate, or to accommodate all users. The policies can be also (sequentially) combined in many different ways; for example, dynamic stream rate selection can be used initially until a lower bound on the stream rate is reached, and then admission control can be used. We evaluate the effectiveness of the policies on a tree-based overlay using experiments that run live code in the ModelNet [27] network emulator that is configured with realistic broadband link characteristics.

The rest of the paper is structured as follows. In Section II we outline the information the overlay needs to expose in order to implement the four adaptation policies. Section III describes the four approaches in detail and Section IV presents evaluation results that demonstrate the successful use of the policies in bandwidth-constrained environment. We discuss related work in Section V and conclude in Section VI.

978-1-4244-5489-1/10/$26.00

II. Information required for adaptation

There are many proposals for streaming overlay protocols [5], [7], [11], [13], [16], [22] which can be broadly split into three categories; unstructured overlays [17], [23], single tree overlays [11], [13], [16] and multiple tree overlays [7], [22], [30]. A recent study comparing resilient overlay multicast approaches can be found in [4].

In a tree, there exists an implicit contract between a parent and a child for a fixed amount of bandwidth to be continuously delivered. In an unstructured overlay, blocks of data arrive from multiple peers at different times which complicates reasoning about available bandwidth. Thus, we examine the adaptation policies in the context of tree-based streaming overlays and leave dealing with unstructured overlays for future work. In order to implement the policies, the overlay needs to expose certain information to the nodes in the overlay. In this section we briefly outline the information required and how this can be collected in a tree-based overlay.

Participating nodes need to be able to estimate their currently available and used outbound bandwidth. As competing flows dynamically change the available bandwidth, we require the estimate to be dynamic. Therefore it is insufficient to use a (often incorrect) bandwidth estimate of the user [10] or to perform one-shot instantaneous measurements of the link capacity at startup. However, dynamically monitoring the available bandwidth can be done by a number of techniques, for example monitoring data packet receive and loss rates for children, and speculatively accepting children. Monitoring data packet deliver and loss rates enables a node to determine whether its outbound link experiences congestion [26]. More elaborate techniques are available which do not require tentatively accepting a child to discover that there is just not enough outbound bandwidth available [15].

We assume that a mechanism exists to generate a summary distribution function of the per-node bandwidth estimates, for both the used outbound bandwidth and the available bandwidth. Further we assume that these are available to all members of the overlay. This can be implemented via compact histograms that are first convergecasted up the tree [20], [22], and then multicasted using the tree. We assume that this information also includes an estimate of the number of nodes in the overlay. Our evaluation shows that the overhead of collecting and distributing this information is low (e.g., 5% of the data stream). In addition, some of the overlays might already be collecting this kind of information to aid in adapting the tree topology to available bandwidth. In our initial design, we assume the information is gathered per each content source. Since users typically view only one stream, we believe this choice is justified.

Finally, for the per-node organizational bandwidth policy we assume that each node is assigned an ID based on the organization to which it belongs, and also the organization's bandwidth limit. We envisage that for small office or home users, this would probably be the ID associated with their ISP. For larger entities, like universities or larger companies, this would be assigned to the entity ID. Members could potentially access the ID and current bandwidth limit using a directory service such as DNS or it can be configured as part of an administrative policy. Organization IDs have to be unique, but it is not necessary to have a global authority assigning them; instead they could be obtained by computing the SHA1 of the IP prefix to assigned to the organization (an organization with multiple IP prefixes can create the ID using just one of the prefixes). This adaptation policy also requires a mechanism for finding parents which satisfy some condition; for this a system like SAAR [20] could be used.

III. Overlay adaptation

Before considering the adaption policies in more detail, we first present a simple model to aid in understanding bandwidth usage for single and multi-tree streaming overlays. First, we assume that the nodes in the overlay are cooperative. Second, let us assume a streaming rate of r kbps, and that the stream can be encoded as z stripes, with $z \geq 1$, such that the bit-rate per stripe is $s = r/z$. If there are N nodes in the overlay (including the source), and each node, j, has an available outbound bandwidth b_j, then j can forward a stripe to $\lfloor b_j/s \rfloor$ children. Therefore, in order to have sufficient forwarding capacity in the overlay, then: $\sum_{j=1}^{N} \lfloor b_j/s \rfloor \geq z(N-1)$ must hold.

More generally, we can define the total number of additional stripes that the overlay can support, C_t, as $C_t = \sum_{j=1}^{N} \lfloor b_j/s \rfloor - kz(N-1)$ where, $k \geq 1$ is a constant that maintains a minimum amount of unused capacity in the overlay. The value of k selected is dependent on the efficiency of the overlay algorithm used. In general, when $C_t > 0$, the overlay is considered to have spare capacity, but when $C_t \leq 0$ then the overlay is considered capacity constrained. The exact point when the overlay has no spare capacity is controlled by k. When $k = 1$ and $C_t < 0$ then the overlay has no spare forwarding capacity and will be unable to forward the stream to all members. Increasing k results in more spare capacity being maintained in the system, making parent discovery easier and the overlay more able to handle churn better. In general, the more efficient the tree building and maintenance protocol is at finding parents when a node is orphaned or joins, the smaller the value of k required.

When $z = 1$ and $C_t \geq 0$ all nodes can receive the stream, but the only topology able to achieve this may be a linear chain topology, as each node may only be able to forward the stream to a single other node. This introduces high latency and low resilience to node churn. Therefore, in order to ensure that a tree is built when $z = 1$, we define the number of additional stripes that the overlay can support, C, as:

$$C = \sum_{j=1}^{N} t(b_j) - kz(N-1) \tag{1}$$

where,

$$t(b) = \begin{cases} \lfloor b/s \rfloor & \text{if } \lfloor b/s \rfloor \geq f, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

The function $t(b)$ returns, for a node's bandwidth, b, the number of children it can support at a stripe rate s, and f captures the minimum bandwidth contribution, in terms of stripes forwarded, that is required for a node to be considered contributing, where $f \geq 1$. For example, to ensure that trees are built when $z = 1$, f should be 2.

Selecting values for k and f carefully is therefore important; k captures the inefficiencies of the overlay protocol at building and maintaining overlays and f captures the relationship between interior nodes and leaf nodes, which results in the formation of efficient distributed tree structures. For example, in the single tree case, when $z = 1$ and $f = 2$, it may be possible to use a value of k close to 1, as many nodes may be able to support a single child, but are therefore excluded from the value of C. In general, the available bandwidth *not* included in C is defined as $\sum_{j=1}^{N} \lfloor b_j/s \rfloor - z(N-1) - C$, and this should always be ≥ 0. This implies that it should be possible to dynamically control k and f across the overlay.

We now will describe each of our overlay adaptation policies in detail and explain their intended use and scope.

A. Node contribution adaptation

There are several scenarios where the bandwidth resources of an overlay can be increased, but at a cost. One such case is where the operator of an overlay has an arrangement with a content distribution network (CDN) to allow the use of the CDN's bandwidth when there is insufficient bandwidth in the overlay. This will (usually) incur a financial cost as the overlay operator will pay the CDN for the bandwidth used. Another example is where the overlay has access to hosts that are currently not viewing the stream, but can be asked to contribute (by requesting the stream and forwarding it to 2 or more nodes), because they are running an instance of the overlay software. We refer to these hosts simply as idle hosts. In this model, the risk is that users may uninstall the software if they perceive it as using excessive bandwidth resources when idle. Therefore, in both approaches it is better to have the CDN or idle hosts only contribute when the overlay needs their resources. It should be noted that both of these approaches have been used in commercially deployed overlays.

In these cases the challenge is to control the contribution of either the CDN or the idle hosts. In both cases these nodes should contribute only when they observe that the there is insufficient aggregate outbound bandwidth in the overlay. Furthermore, we would like to ensure a similar contribution level for each of the idle hosts. We refer to this as controlling the *contribution skew*.

In an overlay without access to a CDN or idle nodes it is also useful to control the contribution skew, to try to ensure a small skew across nodes. This provides fairness, where resources are contributed more evenly across members of the overlay. This is also advantageous because it increases the resilience of the overlay to churn; the more outbound bandwidth a single host contributes, the greater the impact on the overlay when the host fails. We now consider this case in more detail.

To implement this conceptually there is a dynamic overlay-wide maximum contribution level, which defines the maximum outbound bandwidth any host should to contribute to the overlay. The value of the maximum contribution level is set such that there is sufficient aggregate outbound bandwidth. Therefore, hosts with the highest outbound bandwidths do not contribute more than other hosts if there is sufficient aggregate bandwidth. However, if the aggregate bandwidth is low, then the maximum contribution level can be increased.

The information on the bandwidth distribution of the overlay allows every node to locally maintain an estimate of the maximum contribution level. Each node constrains the number of children and the available bandwidth it advertises to ensure that it does not go above the current maximum contribution level. Periodically, each node calculates C to check whether $C \leq 0$. If so, and the node has available outbound bandwidth, it makes a biased, probabilistic decision whether to increase its local maximum contribution level and with it the amount of bandwidth it is willing to contribute. The bias is proportional to the ratio between the host's available bandwidth not advertised (due to the contribution limit) and the sum of available bandwidth not advertised in the complete overlay. The former value is readily available to every host, whereas the later value can be calculated using the distribution functions of total and available bandwidth. Likewise, in case there is enough bandwidth available in the overlay, i.e. $C > 0$, the nodes perform a probabilistic decision to decrease their contribution level.

Thus, while $C < 0$ holds, nodes will increase their contribution until $C \geq 0$, where nodes which contribute less than average increase their contribution with higher probability than nodes which contribute more than the average. This leads to a fair and even distribution of contribution.

B. Dynamic stream rate selection

The goal of overlay adaptation is to satisfy $C > 0$ at all times, and this can be achieved by controlling r. From Equation 1 it can be seen that if the stream publisher can encode the stream at various bit rates, it can change r, thereby influencing the value of C. If $C \leq 0$ then decreasing r will increase C; likewise, if $C > 0$, the increasing r will reduce C, but provide a higher quality stream. The challenge when performing stream rate selection is to select the highest rate r for which $C \geq 0$ will still be satisfied. Using the distribution of total outbound bandwidths in the overlay, the source can periodically perform a binary or brute force search over the set of values encoding rates it can support, to determine the highest value of r such that $C \geq 0$.

Since frequent switching of the stream rate might reduce the perceived video quality, the source can employ some of the standard techniques (e.g., hysteresis) to reduce oscillations.

C. Admission control

The goal of admission control is to explicitly stop nodes from joining the overlay when there is insufficient bandwidth to support them. We assume that using its available bandwidth estimate, a node does not accept more children than it can support. Admission control deals with a similar problem on an overlay-wide level: In many overlays, especially those exploiting multiple trees, a node can join a subset of the trees and consume resources in those trees, but be unable to connect to all required trees. Explicitly telling nodes that they cannot join ensures that joining nodes do not consume resources unless they will be able to join the overlay.

Global admission control is performed as follows. When a node Q wishes to join, it contacts an existing member of the overlay, P, and it passes the initial estimate of its outbound bandwidth. Node P verifies that $C \geq 0$ and, if so, the overlay parent discovery process is used. If not, then P checks to see if Q can contribute the equivalent of w stripes, for example $w \geq 2z$, therefore increasing the likelihood that the joining node contributes more outbound bandwidth than it consumes. If so, the overlay parent discovery process is used, otherwise the node's join request is explicitly rejected. In contrast to the dynamic stream rate selection, which is performed by the source, the admission control check can be performed by any node in the overlay.

D. Per-organization bandwidth limitation

This policy allows a network administrator to limit the amount of bandwidth an overlay will consume at the level of an organization. As organizations are often charged based on the 95th percentile of their consumed bandwidth, this policy can help manage network costs. The limit is enforced across all nodes that belong to an organization, even if it is spanning multiple ISPs, due to multi-homing, wide-area presence, etc. The overlay needs to *i)* enforce this limit, and *ii)* perform locality-aware overlay construction to maximize the number of nodes that it can support.

For the description of this policy and its evaluation, without loss of generality, we only consider the outbound bandwidth of an organization. This policy can be used to limit the inbound bandwidth consumption as well.

In the scope of this policy, we define bandwidth consumption as the amount of data crossing the boundary between an organization and the rest of the world. Specifically, communication between nodes which share a particular organization ID will not contribute to the bandwidth consumption of the organization. If this property does not fit into the administrative structure of an organization, for instance because multiple sites of a company are connected via volume-billed VPN links, it is necessary to assign different IDs to nodes at different sites.

We will now describe how a per-organization bandwidth limit can be implemented. First, we describe the general concept, which we refine subsequently.

Each node acting as a parent records the ID of each child allowing each node to audit how much of its outbound bandwidth is used inter- and intra-organization. Data sent to children with a different ID to the parent contributes to the inter-organization bandwidth usage, whereas children which share their parent's ID contribute to the intra-organization bandwidth. To determine the current inter-organization bandwidth usage of an organization it is necessary to aggregate the information from every node in the organization. To this end, we use a separate control tree for each organization in which the information on bandwidth consumption gets aggregated up and multicasted down.

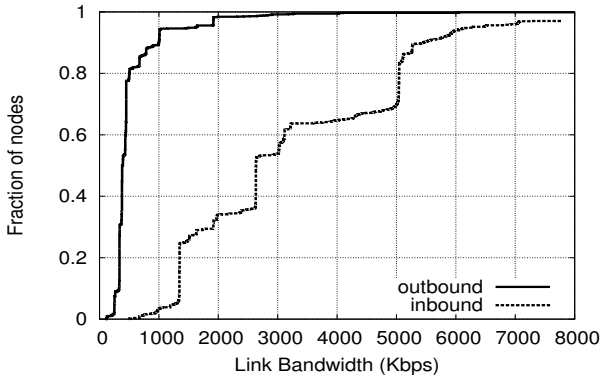

Fig. 1: Distribution of broadband link capacities used in the evaluation setup.

When a node joins the overlay, it first finds a prospective parent, and passes it's ID to that node. If the ID differs from the parent's, then aggregated bandwidth statistics are checked, and if accepting the child will exceed the organization's bandwidth limit the child is told to find another parent. If this is the case, the child is taken on.

In this scenario a preferred parent is therefore one that has the same ID as the joining node. It is therefore preferential to use an organization-aware join, where a node tries to locate a possible parent node in the same organization, for example by using the organization control tree.

IV. Evaluation

All of the experiments use the ModelNet [27] network emulator that lets us run live code. ModelNet routes packets from the end systems through an emulator responsible for accurately emulating the hop-by-hop delay, bandwidth, and congestion of a given network topology; a 3.4-Ghz Pentium-4 running FreeBSD 4.9 served as the emulator for these experiments. We multiplex 350 logical end peers running our application across 13 dual-processor 3.4-Ghz Pentium-4 machines running Linux 2.6.17. All machines are interconnected by a full-rate 1-Gbps Ethernet switch.

We use a 5,000-node INET [8] topology that we further annotate with bandwidth capacities for each link. We keep the latencies generated by the topology generator; the average network RTT is 120ms. We randomly assign our participant peers to act as clients connected to one-degree stub nodes in the topology. We set all links except the client-stub (access) links to be 150 Mbps.

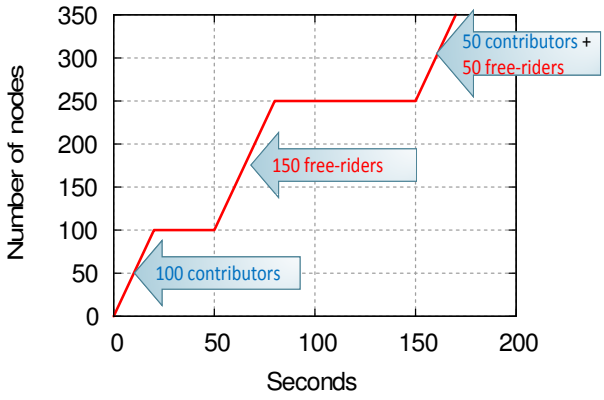

Fig. 2: Depiction of the flash crowd scenario, showing the number of nodes that attempt to join over time.

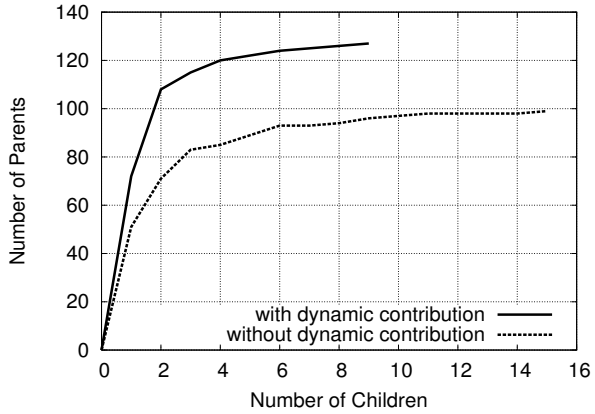

Fig. 3: Per-node number of children CDF with and without node contribution adaptation. Our policy decreases the skew.

We used ModelNet, rather than PlanetLab, in order to use representative measurements of residential broadband networks [12] to set access link capacities (see Figure 1). Specifically, we chose 350 broadband hosts uniformly at random from the entire dataset. For each chosen broadband host, we then configured one participant in our topology with the corresponding broadband host's inbound and outbound capacities. We randomly select one of these participants to act as the media source and set its outbound bandwidth to 5 Mbps. In all the experiments we use a stream rate of 176 Kbps, which is 40 packets per second of 550 bytes.

To evaluate our policies, we subject the streaming overlay to a *flash crowd scenario* (Figure 2). Each node is characterized as being a *contributing* or a *freeriding* node. The contributing nodes are configured to make their outbound bandwidth available to the overlay, while the freeriding nodes are configured to send only control traffic and to not contribute bandwidth to stream data packets. The first flash crowd starts at time 0 seconds and it consists of 100 contributing nodes that attempt to join the overlay at a rate of 5 per second. Fifty seconds into the experiment, we start the second flash crowd of 150 freeriders, again joining at a rate of 5 nodes per second. At 80 seconds into the experiment all 250 nodes have attempted to join the overlay, 100 from the first flash crowd and 150 in the second. Finally, at 150 seconds, the third flash crowd occurs. Another 50 contributing and 50 freeriding nodes join, at a rate of 5 per second.

We evaluate our policies using a single tree overlay, as we intend to demonstrate the effectiveness of the policies, rather than that of any tree maintenance algorithm. Since most of the policies have conflicting approaches to adapting to available bandwidth, we do not attempt to evaluate them one against the other.

As is typical with streaming overlays, the primary performance metric is the rate of packets delivered to the application. Modern codecs can typically deal with small loss (2 to 3 percent [6]) by leveraging the built-in redundancy. Thus, we deem the performance of the overlay satisfactory if up to 97% of packets are delivered to application every second at the default rate (up to 1 packet lost per second).

Individual policies have additional metrics. For example, node contribution adaptation might seek to increase the number of nodes that are contributing, while reducing the variance of individual contributions.

A. Node contribution adaptation

In this experiment, we evaluate the effectiveness of allowing nodes to dynamically scale their contribution to the overlay. To evaluate this policy we modified the underlying network topology such that 14% of the nodes, selected at random, have a symmetric capacity of 5 Mbps inbound/outbound, with all other node capacities unchanged. We ran an experiment using the flash crowd scenario with a single tree overlay, with $k = 1.1$ and $f = 2$. We statically configured all nodes to have an initial maximum contribution-level of two children, independent of their outbound capacity. As described in Section III-A, when a node observes that the spare capacity, C, has fallen below zero, and it has unadvertised capacity it probabilistically increases its maximum contribution-level by one, making it available to the overlay. In order to compare the impact of running with and without the node contribution adaptation we also ran the experiment with contribution adaptation disabled, so all contributor nodes will make their full outbound bandwidth available to the overlay.

The control and probing overhead is low. Across all nodes, probing traffic is 3% of the stream data traffic. The overlay control traffic, consisting primarily of bandwidth histograms being aggregated and disseminated, is only 5% the stream data traffic.

Figure 3 shows the CDF of nodes acting as parents versus the number of children that each parent has for both with and without contribution adaptation. Contribution adaptation is aimed at reducing the skew in contribution across nodes. Without dynamic contribution, just one hundred of nodes act as parents, and 10% of them have more than 6 children. The nodes with 5 Mbps link capacities can support a maximum of

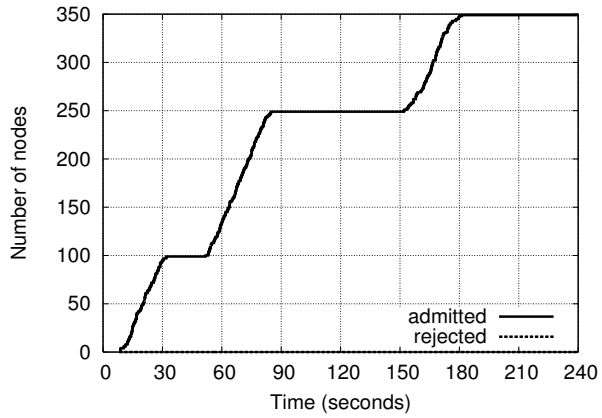

Fig. 4: Overlay membership with node contribution adaptation. All nodes join.

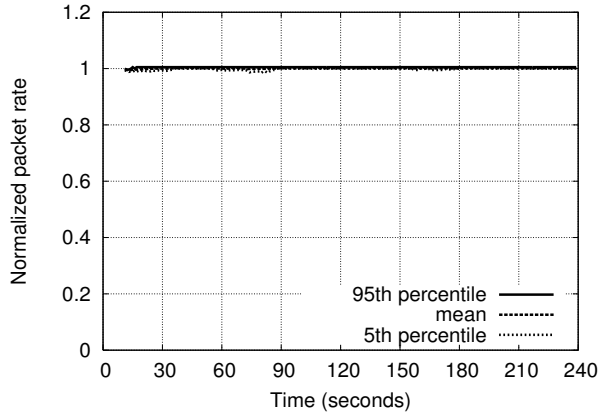

Fig. 5: Packet reception rate with node contribution adaptation.

25 children. In contrast, with dynamic contribution enabled, approximately 25% more nodes capable of supporting children act as parents. Furthermore, only 5 nodes have more than 5 children and the maximum any node has is 9 children. This shows that contribution adaptation is effective.

Figure 4 shows that node contribution adaptation, in addition to smoothing the contribution skew, can also successfully allow all the nodes to join, as would be expected. Each time the overlay has too little spare capacity nodes in the overlay make more capacity available to the overlay. This is demonstrated in Figure 6, which shows how the capacity observed at the histogram at the root varies over time, in terms of number of children that the overlay can support, or *spare slots*. We see that this policy creates approximately 14% spare capacity (at 240 seconds there are 49 spare slots in an overlay of 350 participants), which is slightly more than the target specified by setting $k = 1.1$, which is 10%. These experiments show that the overlay is able to dynamically control the per-node contribution effectively.

B. Admission control

This experiment demonstrates that the overlay can effectively control its membership even when capacity is decreasing

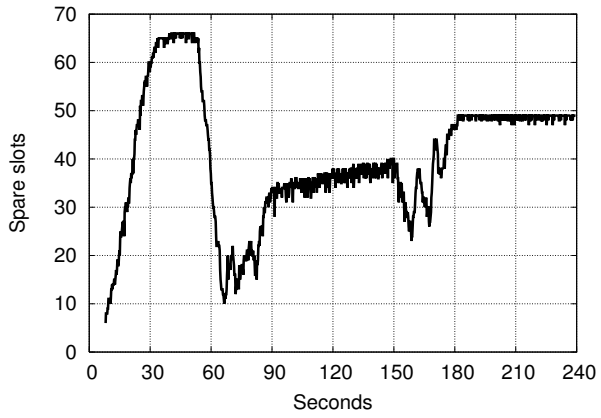

Fig. 6: Overlay capacity (number of spare slots) with node contribution adaptation. All nodes can achieve very good playback quality.

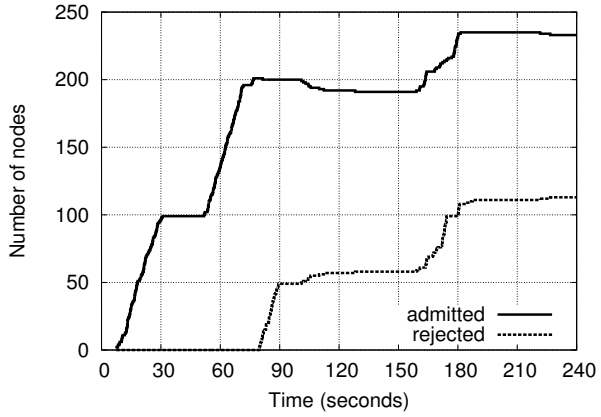

Fig. 7: Overlay membership with the admission control policy under TCP cross-traffic. Clients that cannot be accommodated are rejected.

due to flash crowds and competing TCP traffic. To achieve this in the experiment, we induce TCP cross-traffic with the flash crowd scenario. At 90 seconds, ten randomly chosen contributor nodes from the first flash crowd initiate TCP uploads to other nodes. We configured admission control to reject joining nodes when there was 5% capacity left in the overlay, i.e. $k = 1.05$, and $f = 2$ in Equation 1.

The experiment also demonstrates that the overlay can effectively control its membership. In the experiment there is insufficient aggregate bandwidth to allow all 350 nodes to join the overlay. Figure 7 shows the number of nodes admitted and rejected against time. A node is deemed to have joined the overlay when it has been accepted as a child. Nodes that are explicitly rejected do not attempt to rejoin the overlay. Figure 7 shows that the initial 100 contributing nodes from the first flash crowd are able to join the overlay. The second flash crowd is entirely composed of freerider nodes, and the overlay is unable to support them all, with 50 of them being explicitly rejected.

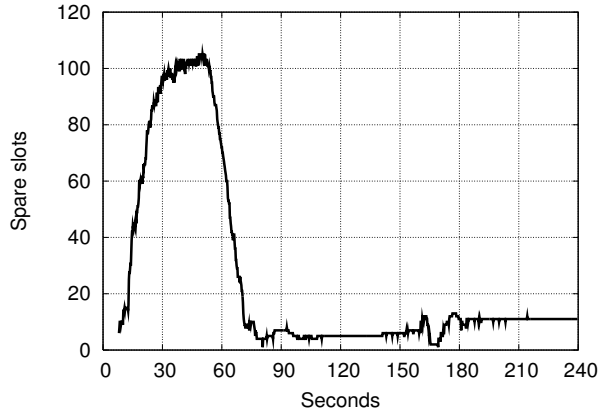

Fig. 8: Overlay capacity (number of spare slots) with the admission control policy under TCP cross-traffic.

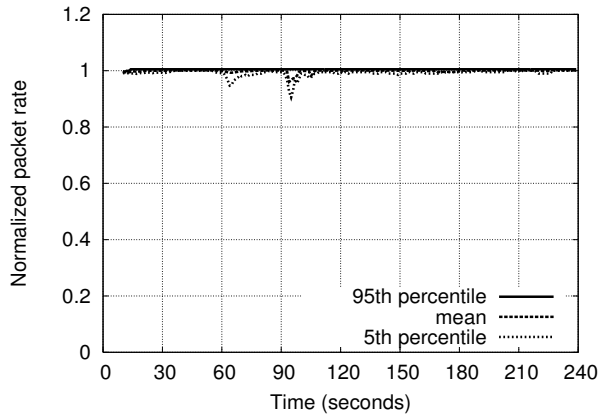

Fig. 9: Packet reception rate with the admission control policy under TCP cross-traffic.

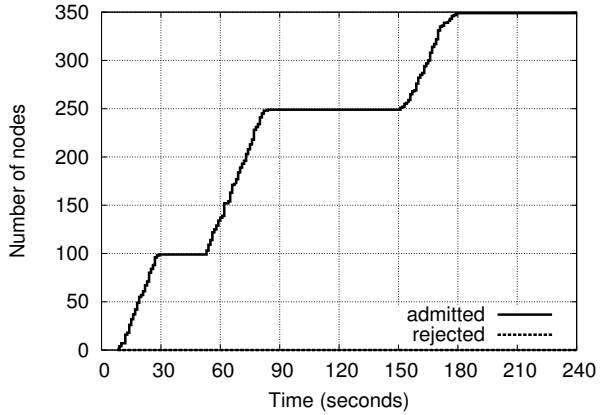

Fig. 10: Overlay membership with the stream rate selection policy. All nodes join.

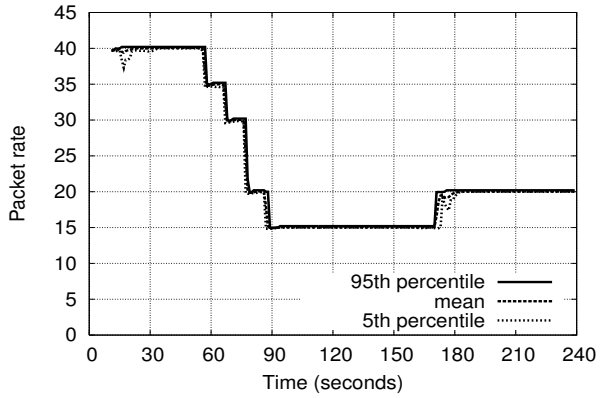

Fig. 11: Packet reception rate with the stream rate selection policy. The source changes the streaming rate to accommodate all nodes.

Figure 7 also shows that at approximately 95 seconds, the overlay membership drops, coinciding with the TCP-flows being initiated. The nodes concurrently serving a TCP flow detect reduced performance among their children and they react by orphaning some of them. Orphaned children try and discover a new parent using the node join process. As the overlay is operating with low spare capacity, the freeriders are unable to rejoin the overlay and are explicitly rejected. This is reflected in Figure 7 which shows an increase in the number of rejected nodes at the same time. Finally, in Figure 7, the third flash crowd introduces new contributing nodes and all but 43 freeriders nodes are accepted.

Figure 8 shows that the number of spare slots increases during the first flash crowd, to about 100. During the flash crowds we are trying to maintain 5% of the slots free, and it can be seen that at approximately 90 seconds the number of free slots dips below this, as some of the spare slots are removed because of the TCP flows. When the final flash crowd occurs, the number of free slots increases to approximately 5% of the total number of slots in the overlay.

Figure 9 shows the mean, 5th and 95th-percentile of the number of packets received by the codec against time. The results show that once a node has joined the overlay it receives a high fraction of the packets, achieving uninterrupted playback. At 90 seconds, the synchronized starting of the TCP flows means that the contributors that that are able to rejoin the overlay experience a short interruption until they locate a new parent.

C. Dynamic stream rate selection

The goal of the this set of experiments is to evaluate the effectiveness of dynamically adjusting the streaming rate. We ran the experiment on an unmodified topology, with $k = 1.2$ and $f = 2$, so the source is trying to maintain at least 20% spare capacity. To achieve this, the source varies the packet rate (the packet size stays constant). We assume that the codec that we are using can generate a packet rate of between 10 and 45 packets per second, in 5 packet intervals. Because the source is changing the streaming rate all nodes are able to join the overlay (Figure 10).

Figure 11 shows the mean and 5th and 95th-percentile of the number of packets received. The first observation is that between 60 and 90 seconds the packet rate drops from 40 packets per second to 15 packets per second. Maintaining 20%

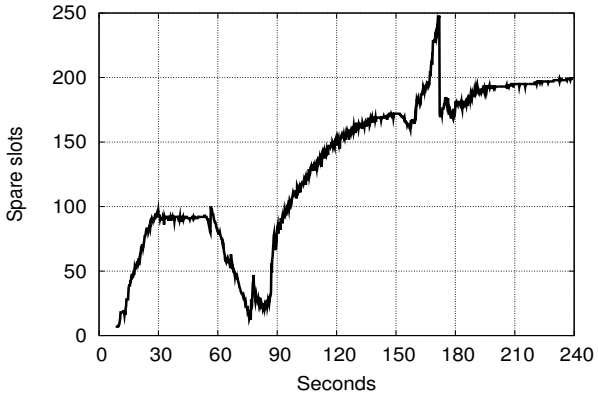

Fig. 12: Overlay capacity (number of spare slots) with the stream rate selection policy.

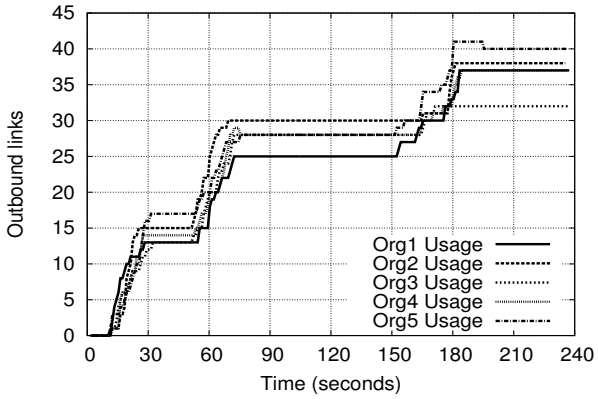

Fig. 13: Per-organization bandwidth consumption (baseline, no limits).

spare capacity means that new nodes can join rapidly, and the source is able to rapidly adapt the streaming rate. Indeed, 95% of nodes join in 10 seconds or less. When the final flash crowd arrives at 170 seconds, the new contributors increase the aggregate bandwidth available. This is sufficient to allow the source to increase the stream rate. After the rate increase, some nodes are no longer capable of supporting all their current children at the new rate. This causes some children to be orphaned, and have to seek a new parent. This is reflected in a slight dip in the 5th percentile packet receive rate.

Finally, Figure 12 shows the spare slots, again using the histogram generated at the root. It should be noted that the spare slots are a function of the packet rate. The overlay capacity is increasing between 90 and 150 seconds because the available per-node bandwidth that was not enough to support a child at a higher rate is enough to create spare slots at a lower rate. At 170 seconds, an additional increase in slots can be clearly seen due to the presence of contributors in final flash crowd joining. The number of slots changes abruptly, as the source changes the stream rate from 15 to 20 packets per second, representing an increase of 25%. The fall in spare slots does not exactly correspond to 25% as some nodes will be unable to contribute as much bandwidth as they were. For example, a node may be able to support 45 packets per second, and have three children when the streaming rate is 15 packets per second. When the streaming rate increases to 20 packets per second, it will only be able to support two children, and hence it will be unable to contribute the remaining 5 packets per second.

D. Per-organization bandwidth limitation

To evaluate per-organization bandwidth limits, we assign every node to one of five existing organizations, thus each organization contains 70 nodes. We establish a baseline case by running a basic admission control scenario and recording the amount of bandwidth used by every organization. Figure 13 shows that in this case every organization accepted about 35 nodes from other organizations while, as expected, the acceptance rate was similar to the one shown for pure admission control. Figure 15 shows that the packet reception rate was acceptable throughout the experiment.

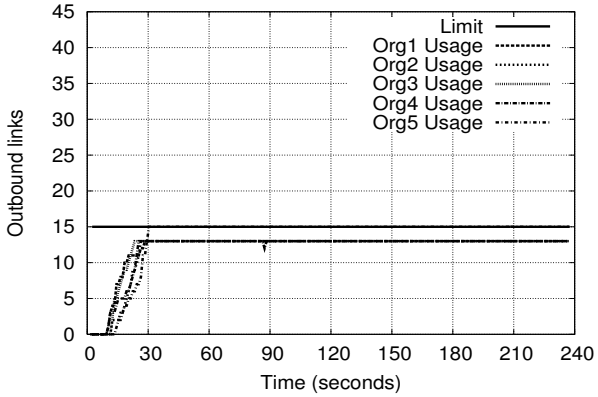

Fig. 14: Per-organization bandwidth consumption with the limits enabled, locality-aware joins disabled.

We then set 15 as the limit for the number of accepted out-of-organization nodes. To demonstrate that we can effectively limit the bandwidth consumption, we first disable the locality-aware joins. As Figure 14 shows, every organization was at or slightly under the proposed limit in this case, the penalty being the 28 percent lower acceptance rate (Figure 16).

However, when a client first tries to accommodate itself within its own organization (which is the intended behavior of this policy), the number of accepted nodes matches the baseline case. In addition, Figure 17, shows that all organizations are well below the bandwidth limit. This demonstrates that per-organization bandwidth limitation with locality-aware joins is efficient in in avoiding inter-organization traffic.

V. Related work

Wu *et al.* [28] characterize the throughput of TCP flows in a large-scale peer-to-peer streaming system, and conclude that most often the throughput constraining factor is the capacity of the last-mile (uplink). Despite the large body of work on overlays, the problem of dealing with available bandwidth scarcity has not received considerable attention.

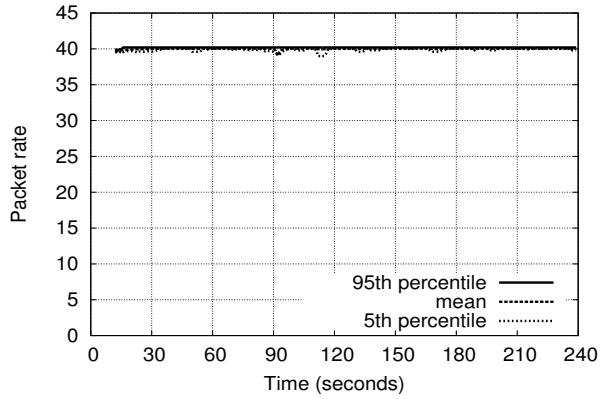

Fig. 15: Packet reception rate with the limits enabled, locality-aware joins disabled.

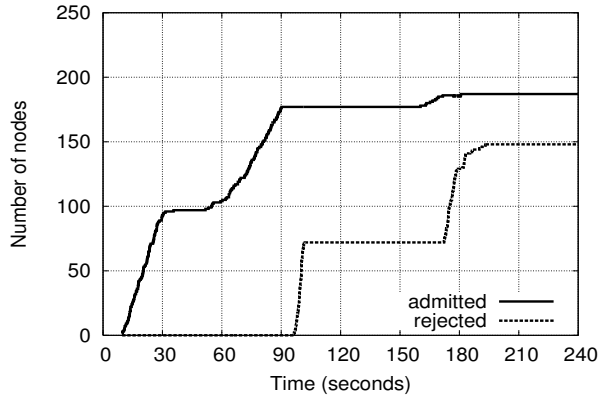

Fig. 16: Overlay membership with per-organization bandwidth limits enabled, locality-aware joins disabled. More nodes are rejected than in the baseline case.

Sripanidkulchai *et al.* [25] produced an analysis of the streaming video traffic distributed using Akamai to determine if it was feasible to distribute the streams using an overlay. They concluded that there was sufficient bandwidth at the hosts to support single tree based overlay distribution. However, concurrent work by Chu *et al.* [10], discussed lessons learnt from deploying a real overlay and observes that in some scenarios they required the use of *waypoints*, which are infrastructure based nodes that provide extra bandwidth to the overlay when the aggregate hosts' outbound bandwidth was insufficient. Our per-node dynamic contribution scaling policy that we describe in Section III-A and evaluate in Section IV-A accomplishes this task.

The most related work for managing bandwidth in overlays is that of Sung *et al.* [26]. They describe a contribution-aware overlay supporting live-streaming for environments with limited and asymmetric bandwidth, and significant heterogeneity in outbound bandwidth. The basic idea is to force participants to contribute more bandwidth than they consume via a fixed taxation rate, thereby ensuring that there is always some extra spare capacity in the system. Peers receive different levels of performance based on their bandwidth contribution. We believe that contribution-awareness and bandwidth adaptation are largely orthogonal topics. As such, we consider our work and [26] as being supplementary to each other. For example, our policies do not provide incentives for contribution, while their taxation system is not powerful enough to enable bandwidth adaptation strategies like CDN usage or dynamic stream rate selection.

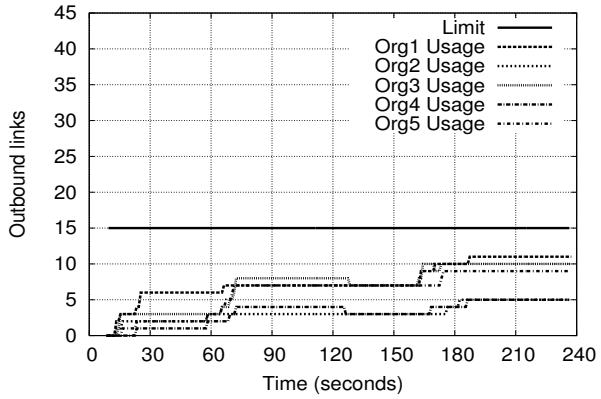

Fig. 17: Per-organization bandwidth consumption with the limits enabled, locality-aware joins enabled.

Several systems have proposed using multiple description codecs (MDCs) or layered encodings (LM) [22], [26] combined with some form of tit-for-tat mechanism [14], [17], [26], which attempts to ensure that hosts contribute as much, or more, bandwidth than they consume. Originally, layered encoding was proposed for use with IP Multicast [19], and was designed to handle heterogeneous *inbound* bandwidth capacities. Today, many hosts have significantly more inbound than outbound bandwidth. The challenge for overlays is therefore to manage the outbound rather than the inbound bandwidth. If the tit-for-tat mechanism is strictly enforced, then outbound bandwidth constrained hosts may not be able to view the stream. Furthermore, a tit-for-tat mechanism implies that the overlay needs to be constructed such that all hosts can contribute. Due to the prevalence of firewalls and NATs this is not always easy to do in reality. The use of tit-for-tat mechanisms may be ideologically attractive, yet from a commercial content provider's perspective, it is undesirable to deny service to a customer because they have little outbound bandwidth if other hosts in the overlay have spare.

A related bandwidth allocation problem is limiting total inbound or outbound bandwidth consumed by an overlay service within an organization. There exist commercial solutions which can limit bandwidth on a per-site basis [21], but these cannot enforce a limit over a set of wide-area network locations. Recently Raghavan *et al.* [24] described methods for controlling the bandwidth consumption of an ensemble of flows which do not have to pass through the same infrastructure. Although these approaches can limit the amount of bandwidth an organization devotes to an overlay service, they are not appropriate for a streaming service as a

large fraction of nodes might experience interrupted playback when rate limiting takes place.

Our work is orthogonal to the work on capacity-aware overlay tree construction algorithms that take into account the heterogeneous nature of access-link capacity [5], [20]. There has also been work showing algorithms that fail to take heterogeneity into account perform badly for high-bandwidth applications [3], [10].

Recently there have been numerous proposals to make overlay systems locality-aware [1], [9]. P4P [29] allows cooperation between overlay applications and ISPs in containing their traffic within the provider networks. PACE [2] lets operators set prices for transit links to provide incentives for nodes to prefer peers which are close. One of our policies goes one step further, in that it provides the network administrator of an organization the ability to specify a bandwidth limit for the overlay traffic. In addition to ensuring good playback quality under the limit, our policy works for organizations that span multiple ISPs.

VI. Conclusion

In this paper we argue for the importance of enabling streaming overlays to successfully adapt to system-wide bandwidth availability. We have described and demonstrated four policies that can each accomplish this task: node contribution adaptation, dynamic stream rate selection, admission control, and per-organization bandwidth limitation. Our live experiments with realistic link capacities demonstrate that all the policies are effective.

It is up to the content provider to choose which policy to apply and when. For example, the provider might want to accept as many paying customers as possible by decreasing the streaming rate if the aggregate outbound bandwidth is at the limit. Once the rate is at the lowest setting supportable by the codec, it might be prudent to reject some incoming users to preserve the viewing quality of existing participants. Another option would be to request from the existing users to contribute more using the node contribution adaptation policy.

Acknowledgments

We would like to thank Antony Rowstron for his contribution to this project. Simon Schubert is supported by a Microsoft Research PhD Scholarship.

References

[1] V. Aggarwal, A. Feldmann, and C. Scheideler. Can ISPs and P2P Users Cooperate for Improved Performance? *SIGCOMM Comput. Commun. Rev.*, 37(3):29–40, 2007.

[2] C. Aperjis, M. J. Freedman, and R. Johari. Peer-Assisted Content Distribution with Prices. In *Proceedings of CoNext*, 2008.

[3] A. Bharambe, S. G. Rao, V. Padmanabhan, S. Seshan, and H. Zhang. The Impact of Heterogeneous Bandwidth Constraints on DHT-Based Multicast Protocols. In *IPTPS*, Feb. 2005.

[4] S. Birrer and F. E. Bustamante. A comparison of resilient overlay multicast approaches. *IEEE JSAC*, 25(9):1695–1705, 2007.

[5] M. Bishop, S. Rao, and K. Sripanidkulchai. Considering Priority in Overlay Multicast Protocols under Heterogeneous Enviroments. In *INFOCOM*, 2006.

[6] F. Bonomi, M. Mitzenmacher, R. Panigrahy, S. Singh, and G. Varghese. Beyond Bloom Filters: From Approximate Membership Checks to Approximate State Machines. In *Proceedings of SIGCOMM*, 2006.

[7] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Multicast in a Cooperative Environment. In *Proceedings of SOSP*, 2003.

[8] H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. Towards Capturing Representative AS-Level Internet Topologies. In *Proceedings of ACM SIGMETRICS*, June 2002.

[9] D. R. Choffnes and F. E. Bustamante. Taming the Torrent: A Practical Approach to Reducing Cross-ISP Traffic in Peer-to-Peer Systems. In *Proceedings of SIGCOMM*, 2008.

[10] 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 *Proceedings of USENIX*, June 2004.

[11] Y. Chu, S. G. Rao, S. Seshan, and H. Zhang. Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture. In *Proceedings of SIGCOMM*, 2001.

[12] M. Dischinger, A. Haeberlen, K. P. Gummadi, and S. Saroiu. Characterizing Residential Broadband Networks. In *Proceedings of the Internet Measurement Conference (IMC)*, 2007.

[13] P. Francis. Yoid: Extending the Internet Multicast Architecture. Technical Report, April 2000.

[14] A. Haeberlen, P. Kouznetsov, and P. Druschel. PeerReview: Practical Accountability for Distributed Systems. In *Proceedings of SOSP*, 2007.

[15] N. Hu and P. Steenkiste. Evaluation and Characterization of Available Bandwidth Probing Techniques. *IEEE Journal on Selected Areas in Communications*, 21(6):879–894, Aug. 2003.

[16] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. James W. O'Toole. Overcast: Reliable Multicasting with an Overlay Network. In *Proceedings of OSDI*, October 2000.

[17] H. Li, A. Clement, M. Marchetti, M. Kapritsos, L. Robinson, L. Alvisi, and M. Dahlin. FlightPath: Obedience vs choice in cooperative services. In *Proceedings of OSDI 2008*, Dec 2008.

[18] J. Liu, S. Rao, B. Li, and H. Zhang. Opportunities and challenges of peer-to-peer internet video broadcast. In *IEEE Special Issue on Recent Advances in Distributed Multimedia Communications*, 2007.

[19] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven Layered Multicast. In *Proceedings of SIGCOMM*, August 1996.

[20] A. Nandi, A. Ganjam, P. Druschel, T. S. E. Ng, I. Stoica, H. Zhang, and B. Bhattacharjee. SAAR: A Shared Control Plane for Overlay Multicast. In *Proceedings of NSDI*, 2007.

[21] Packeteer. http://www.packeteer.com.

[22] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. Resilient Peer-to-Peer Streaming. In *Proceedings of ICNP*, Atlanta, Georgia, USA, 2003.

[23] PPLive. PPLive web site. http://www.pplive.com.

[24] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. Snoeren. Cloud Control with Distributed Rate Limiting. In *Proceedings of SIGCOMM*, 2007.

[25] K. Sripanidkulchai, B. Maggs, and H. Zhang. The Feasibility of Supporting Large-Scale Live Streaming Applications with Dynamic Application End-Points. In *Proceedings of ACM SIGCOMM*, Aug. 2004.

[26] Y. W. E. Sung, M. A. Bishop, and S. G. Rao. Enabling Contribution Awareness in an Overlay Broadcasting System. *IEEE Transactions on Multimedia*, 9(8):1605–1620, Dec. 2007.

[27] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostić, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. In *Proceedings of the 5th OSDI*, 2002.

[28] C. Wu, B. Li, and S. Zhao. Characterizing peer-to-peer streaming flows. *IEEE JSAC*, 25(9):1612–1626, 2007.

[29] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. Liu, and A. Silberschatz. P4P: Provider Portal for Applications. In *Proceedings of SIGCOMM*, 2008.

[30] S. Xie, B. Li, G. Y. Keung, and X. Zhang. Coolstreaming: Design, Theory, and Practice. *IEEE Transactions on Multimedia*, 9(8):1661–1671, Dec. 2007.