Enabling Conferencing Applications on the Internet using an Overlay Multicast Architecture *

Yang-hua Chu, Sanjay G. Rao, Srinivasan Seshan and Hui Zhang

Carnegie Mellon University

{yhchu, sanjay, srini+, hzhang}@cs.cmu.edu

ABSTRACT

In response to the serious scalability and deployment concerns with IP Multicast, we and other researchers have advocated an alternate architecture for supporting group communication applications over the Internet where all multicast functionality is pushed to the edge. We refer to such an architecture as End System Multicast. While End System Multicast has several potential advantages, a key concern is the performance penalty associated with such a design. While preliminary simulation results conducted in static environments are promising, they have yet to consider the challenging performance requirements of real world applications in a dynamic and heterogeneous Internet environment.

In this paper, we explore how Internet environments and application requirements can influence End System Multicast design. We explore these issues in the context of audio and video conferencing: an important class of applications with stringent performance requirements. We conduct an extensive evaluation study of schemes for constructing overlay networks on a wide-area test-bed of about twenty hosts distributed around the Internet. Our results demonstrate that it is important to adapt to both latency and bandwidth while constructing overlays optimized for conferencing applications. Further, when relatively simple techniques are incorporated into current self-organizing protocols to enable dynamic adaptation to latency and bandwidth, the performance benefits are significant. Our results indicate that End System Multicast is a promising architecture for enabling performance-demanding conferencing applications in a dynamic and heterogeneous Internet environment.

*This research was sponsored by DARPA under contract number F30602-99-1-0518, and by NSF under grant numbers Career Award NCR-9624979 ANI-9730105, ITR Award ANI-0085920, and ANI-9814929. Additional support was provided by Intel. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of DARPA, NSF, Intel or the U.S. government.

SIGCOMM'01, August 27-31, 2001, San Diego, California, USA..

1. INTRODUCTION

Over the last decade, researchers have studied how group communication applications like audio and video conferencing, multi-party games, content distribution, and broadcasting can be supported using IP Multicast[4]. However, over ten years after its initial proposal, IP Multicast is yet to be widely deployed due to fundamental concerns related to scalability, and support for higher layer functionality like reliability and congestion control. Recently, there has been a reevaluation by the research community of whether IP is indeed the right layer to support multicast-routing related functionality. A growing number of researchers [2, 3, 6, 9] have advocated an alternate architecture, where all multicast related functionality, including group management and packet replication, is implemented at end systems. We refer to such an architecture as End System Multicast. In this architecture, end systems participating in a multicast group self-organize into an overlay structure using a completely distributed protocol. Further, end systems attempt to optimize the efficiency of the overlay by adapting to network dynamics and considering application level performance.

While End System Multicast can have several potential advantages, a key concern is the performance of such an approach. While several recent works have demonstrated that the performance penalty of using overlays can be acceptably low, these studies have been conducted primarily using simulation experiments, static metrics and controlled environments [2, 3, 9]. However, Internet environments, the target of these overlay architectures, are dynamic, heterogeneous and unpredictable. In this paper we focus on a key question regarding the feasibility of an overlay architecture: can such an architecture satisfy the demanding end-to-end performance requirements of real world applications in such an environment?

We study this question in the context of an important class of applications: audio and video conferencing. Internet based conferencing applications have received a great amount of attention in the last decade, during which excellent tools like *vic*[10], *vat*[8] and *rat*[7] were developed. Yet, these tools are not ubiquitously deployed today due to the limited availability of IP Multicast. Conferencing applications have stringent performance requirements, and are among the most challenging to support. They not only require a high sustained throughput between the source and receivers, but also require low latencies.

In order to meet these performance requirements, we show that it is necessary for self-organizing protocols to adapt to both latency and bandwidth metrics. We present techniques by which such protocols can adapt to dynamic metrics like available bandwidth and latency, and yet remain resilient to

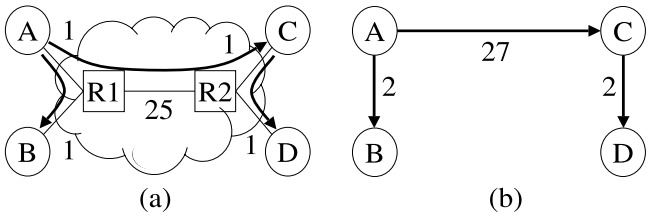

Figure 1: Example of End System Multicast

network noise and inaccuracies inherent in the measurement of these quantities. We demonstrate our ideas by incorporating them into Narada, a self-organizing protocol that we presented in [3]. While we have chosen to use Narada, we believe that the techniques we present can be incorporated into other self-organizing protocols. In addition, although our techniques take advantage of some of the unique characteristics of conferencing applications, we believe that they could benefit other classes of group communication applications as well.

We evaluate our techniques by testing the redesigned Narada protocol on a wide-area test-bed. Our test-bed comprises twenty machines that are distributed around North America, Asia and Europe. Our results demonstrate that our techniques can provide good performance, both from the application perspective and from the network perspective. With our scheme, the end-to-end bandwidth and latency attained by each receiver along the overlay is comparable to the bandwidth and latency of the unicast path from the source to that receiver. Further, when our techniques are incorporated into Narada, applications can see improvements of over 30–40% in both throughput, and latency. Finally, the costs of our approach can be restricted to 10–15% for groups of up to twenty members.

The rest of the paper is organized as follows. We begin by providing an overview of End System Multicast and self-organizing protocols in Section 2. Section 3 presents important performance issues that self-organizing protocols need to address to support conferencing applications. Our techniques for tackling these issues are presented in Section 4. Sections 5, 6 and 7 present our evaluation methodology and results. Finally, we discuss our results, present related work, and summarize in Sections 8, 9, and 10.

2. END SYSTEM MULTICAST

In End System Multicast, nodes participating in a multicast group, or proxies that operate on their behalf, organize themselves into overlay spanning trees for data delivery. The tree is an overlay in the sense that each link corresponds to an unicast path between two end systems in the underlying Internet. For instance, consider Figure 1(a), which depicts a physical network where $R1$ and $R2$ are routers, and A, B, C and D are end systems. Figure 1(b) shows an overlay tree rooted at source A, while Figure 1(a) also shows how this overlay tree maps on to the underlying physical network.

End System Multicast occurs in two distinct architectural flavors: peer-to-peer architectures and proxy based architectures. In the former, all functionality is pushed to the end hosts actually participating in the multicast group. In a proxy based architecture on the other hand, an organization that provides value added services deploys proxies at strategic locations on the Internet. End hosts attach themselves to proxies near them and receive data using plain unicast.

The End System Multicast architecture provides several advantages over IP Multicast. First, it requires absolutely no network level support for multicast, and all multicast functionality is pushed to the edge. By avoiding per-group state in routers, the inherent scaling concerns introduced by IP Multicast are avoided. Further, deployment is not an issue, as no change is required to network infrastructure. Finally, we believe that solutions for supporting higher layer features such as error, flow and congestion control can be significantly simplified by deploying application intelligence at internal splitting points of an overlay tree.

While End System Multicast has several advantages, a fundamental challenge is providing a method for nodes to self-organize into an overlay network that efficiently forwards multicast packets. There has been a spurt of activity in the design of self-organizing protocols for End System Multicast in the last year [2, 3, 6, 9]. These self-organizing protocols primarily consist of two components: (i) a group management component; and (ii) an overlay optimization component. The group management component ensures that the overlay remains connected in the face of dynamic group membership and failure of members. The overlay optimization component ensures that the quality of the overlay remains good over time. Overlay optimization involves obtaining network information using a variety of techniques such as active measurements and passive monitoring of performance. As more information is available about the network, or as network conditions change, the overlay can be modified by the addition of good links and the dropping of poor links.

Two basic methods have emerged for the construction of overlay spanning trees for data delivery. One approach is to construct the tree directly - that is, members explicitly select their parents from among the members they know. Yoid [6] and Overcast [9] adopt this approach. An alternate approach which protocols such as Gossamer [2] and Narada [3] use, is to construct trees in a two-step process. First they construct a richer connected graph termed a *mesh*. Second, they construct (reverse) shortest path spanning trees of the mesh, each tree rooted at the corresponding source using well-known routing algorithms. While in the extreme case, a mesh could consist of all possible $N*(N-1)$ overlay links in a group consisting of N members, typically protocols try to keep the meshes sparse in order to keep the overhead of routing low. Given that the final spanning trees are constructed from among the overlay links present in the mesh, it becomes important to construct a good quality mesh in the first place. The reader is referred to [3, 6] for further discussion of the tree-first and mesh-first approaches.

3. CONFERENCING APPLICATIONS AND OVERLAY DESIGN

A key feature of End System Multicast is that it enables application customizable protocols and architectural decisions. In this section, we study the interaction between End System Multicast and conferencing applications. We begin by reviewing the following distinguishing characteristics of conferencing applications:

- *Performance requirements:* Conferencing applications require low latencies, and need to sustain high bandwidth between the source and receivers. In contrast, broadcasting and file transfer applications are primarily interested in bandwidth, and latency is not a concern.
- *Gracefully degradable:* Conferencing applications deal with

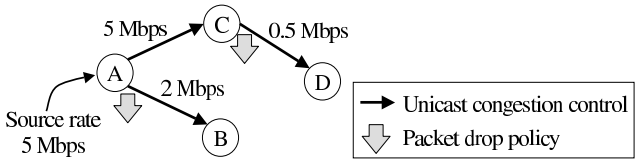

Figure 2: Architectural framework for supporting conferencing applications

media streams that can tolerate loss through a degradation in application quality. This is in contrast to file transfer applications that require reliable data delivery.

• *Session lengths:* Conferences are generally long lived, lasting tens of minutes. In contrast, applications like file transfer and software downloading may be short-lived, lasting for the duration of the transfer.

• *Group characteristics:* Conferences usually involve small groups, consisting of tens to hundreds of participants. Membership can be dynamic. Again, this is in contrast to applications like broadcasting, and content delivery that may deal with much larger group sizes.

• *Source transmission patterns:* Typically, conferencing applications have a source that transmits data at a fixed rate. While any member can be the source, there is usually a single source at any point in time. In contrast, large scale broadcasting applications have a single static source throughout a session.

Several conferencing application features are well suited to existing End System Multicast techniques. For example, self-organizing protocols employ self-improving algorithms, and incrementally produce better overlays by learning network path characteristics and adapting to network dynamics. The small group sizes and long session durations of conferences match such an approach.

Some aspects of conferencing applications enable relatively straight-forward application-specific solutions to existing problems. For example, the graceful degradation of media streams allows us to build a system that employs a hop-by-hop congestion control protocol. Congestion control on each individual overlay link is ensured by running some TCP-friendly protocol for streaming media applications [1, 5, 14]. An overlay node adapts to a bandwidth mismatch between the upstream and downstream links by dropping packets. Figure 2 shows an example of an overlay tree, where A is the source. Links A-B and C-D cannot sustain the source rate of 5 Mbps, and consequently nodes A and C reduce the rate using some appropriate packet drop policy.

The performance requirements of conferencing applications are one key aspect that existing End System Multicast systems cannot support. In this paper, we focus on addressing this key issue by incorporating techniques in self-organizing protocols to support the following:

• *Optimizing for dual metrics:* Overlay links need to be chosen in such a manner as to simultaneously ensure high bandwidth and low latencies from every source to each receiver.

• *Optimizing for dynamic metrics:* Internet latencies and available bandwidth are dynamic, and the overlay needs to adapt to long-term variations in path characteristics. Yet, it needs to be resilient to network noise and inaccuracies that is inherent in the measurement of these quantities. Frequent changes to overlay topology could result in instability and transient performance degradation.

4. CONFERENCING OPTIMIZED OVERLAYS

In this section, we present a set of techniques that help self-organizing protocols deal with the challenges of supporting conferencing applications. While we believe our ideas can easily be incorporated into all End System Multicast protocols, we choose to demonstrate them on the Narada protocol [3]. Narada is a mesh-based protocol, and runs a distance vector algorithm extended with path information on top of the mesh. It leverages a DVMRP-like algorithm for constructing the spanning trees for data delivery. Further details of Narada can be found in [3].

4.1 Dealing with dual and dynamic metrics

Constructing an overlay optimized for both latency and bandwidth can be difficult. In designing heuristics for tackling this problem, we have been motivated by the work done by Wang and Crowcroft [15] in the context of routing on multiple metrics in the Internet. A first choice is to optimize the overlay for a single mixed metric that is a function of both bandwidth and latency. However, it is not clear how this function can individually reflect the bandwidth and latency requirements of the application. A second approach is to treat the two metrics explicitly and with equal importance. Thus, a change would be made to the overlay if either the bandwidth or the latency improves as a result of that change. However, this approach could lead to oscillations when confronted with two conflicting options, one with better latency, and the other with better bandwidth but poorer latency. Instead, we consider both bandwidth and latency explicitly, but prioritize bandwidth over latency. We believe that this prioritization reflects the application semantics better.

We incorporate this idea in Narada, by choosing multiple routing metrics in the distance vector protocol running on the mesh - the *available bandwidth* and the *latency* of the overlay link. The routing protocol uses a variant of the *shortest widest path* algorithm presented in [15]. Every member tries to pick the *widest (highest bandwidth)* path to every other member. If there are multiple paths with the same bandwidth, the member picks the *shortest (lowest latency)* path among all these.

Both available bandwidth and latency are dynamic in nature, and using them as routing metrics leads to serious concerns of instability. We deal with the stability concerns using techniques in the design of the routing metrics described below:

• *Latency:* We filter raw estimates of the overlay link latency using an exponential smoothing algorithm. The advertised link latency is left unchanged until the smoothed estimate differs from the currently advertised latency by a significant amount.

• *Available bandwidth:* We filter raw estimates of the available bandwidth of an overlay link using an exponential smoothing algorithm, to produce a *smoothed estimate*. Next, instead of using the smoothed estimate as a routing metric, we define discretized bandwidth levels. The smoothed estimate is rounded down to the nearest bandwidth level for routing purposes. Thus, a mesh link with a smoothed estimate of 600 Kbps may be advertised as having a bandwidth of 512 Kbps, in a system with levels corresponding to 512 Kbps and 1024 Kbps. To tackle possible oscillations if the smoothed estimate is close to a bandwidth level, we employ a simple hysteresis algorithm. Thus, while we move down a level immediately when the smoothed estimate falls below the current level, we move up a level only if the esti-

mate significantly exceeds the bandwidth corresponding to the next level.

Given that conferencing applications often have a fixed source rate, the largest level in the system is set to the source rate. Discretization of bandwidth and choice of a maximum bandwidth level ensure that all overlay links can fall in a small set of equivalence classes with regard to bandwidth. This discretized bandwidth metric not only enables greater stability in routing on the overlays, but also allows latency to become a determining factor when different links have similar but not identical bandwidth.

Given a good quality mesh, the mechanisms described above seek to construct overlay trees that ensure good bandwidth and latencies between every source and the recipients. We retain the basic mechanisms presented in the Narada protocol to improve the quality of the mesh itself. Members probe non-neighbors at random, and may add a new link to the mesh if the utility gain of adding the link exceeds a threshold. Members monitor existing links, and drop them if the cost of dropping the link falls below a threshold. The utility gain, and cost are computed based on the number of members to which performance improves (degrades) in bandwidth and latency if the mesh link were added (dropped), and the significance of the improvement (degradation).

4.2 Metric Estimation

In this section, we present details of how we collect raw estimates of the latency and bandwidth of overlay links. These estimates are used by the routing algorithms presented in Section 4.1. We use different mechanisms for collecting bandwidth and latency estimates for links in the mesh, and for links that are not.

Members determine latencies of links in the mesh by periodically (currently every 200 milliseconds) exchanging packets with their neighbors and estimating the round trip time. The link latency is assumed to be half the round trip time. While these measurements turn out to have a low overhead, we note that when the underlying transport protocol allows, and there is data flow along a mesh link, we may directly obtain round trip time estimates by querying the transport protocol.

We keep bandwidth estimates of links already in the mesh up to date by passively monitoring the performance of these links when there is data flow along them. Members periodically advertise the rates at which they are transferring data to their neighbors along a mesh link. The neighbor compares this advertised estimate, with an estimate of data it actually receives along that mesh link. If the rates are comparable, it treats the estimate as a lower bound on available bandwidth. Otherwise, it assumes the rate at which it receives data is an actual estimate of the bandwidth of the link.

Bandwidth estimates of links not in the mesh are currently determined using active end-to-end measurements, which involves transferring data using the underlying transport protocol for 15 seconds, but at a rate bounded by the maximum source rate. As active measurements can have a high overhead, we have adopted simple techniques to minimize the number of such measurements:

• If a member is receiving poor performance because of congestion on its local access link (for example, a machine behind ADSL, or a modem), this member does not probe any other member. We currently use a simple heuristic to determine congestion on a local link: we *ping* the first hop router on the member's path to the Internet, and determine the local link to be congested if the ping times exceed a threshold. We have found this heuristic to work reasonably well in many situations.
• Member A conducts an active bandwidth measurement to member B, only if B itself gets good performance from other members, and has the potential to improve A's performance. A determines the quality of B's performance to other members by examining its routing table which it obtains using a small number of message exchanges. We find this mechanism helpful in heterogeneous environments, where a good member can avoid probing a member behind a modem or ADSL connection. Further, it prevents members who are already doing well in the overlay from probing other members.

Bandwidth estimates may be outdated due to a change of network conditions since the last estimate was made, or inaccurate due to noise inherent in these measurements. To keep estimates to other members timely, a member may probe another member for which there has been no bandwidth estimate for an extended period of time. Currently, a member may conduct an active bandwidth measurement to a mesh neighbor for which there has been no bandwidth estimate in the last five minutes, and to a non-neighbor for which there has been no bandwidth estimate in the last twenty minutes.

5. EXPERIMENTAL EVALUATION

Our evaluation seeks to answer the following questions:

• From the application perspective, can End System Multicast meet the bandwidth and latency requirements of conferencing applications in the Internet?
• How critical is it to adapt to network performance metrics such as bandwidth and latency while constructing overlays?
• What are the network costs and overheads associated with the self-organizing overlay architectures we consider?

To answer these questions, we examine the performance of several schemes for constructing overlay networks, described in Section 5.1. Of these schemes, ours is the only one that adapts dynamically to both bandwidth and latency. All other schemes consider only one of these metrics, or none at all. Section 6 presents detailed results that compare the performance of our scheme with all other schemes. In Section 7, we focus on our scheme and analyze several aspects pertaining to how it adapts to congestion in the network.

Two important factors affect the performance of a scheme for constructing overlay networks. These critical factors are the characteristics of the source application and the degree of heterogeneity in the host set we consider. Less demanding applications and more homogeneous environments can make even a poorly constructed overlay perform adequately.

We consider the performance of the schemes with different speed constant bit rate (CBR) sources. CBR encodings are common in conferencing applications, and make our evaluation convenient. To study the performance of overlay schemes in environments with different degrees of heterogeneity, we create two groupings of hosts, the *Primary Set* and the *Extended Set*. The *Primary Set* contains 13 hosts located at university sites in North America where nodes are in general well-connected to each other. The *Extended Set* contains 20 hosts, and includes a machine behind ADSL, and hosts in Asia and Europe, in addition to the hosts in

the *Primary Set*. Thus, there is a much greater degree of variation in bandwidth and latencies of paths between nodes in the *Extended Set*.

We conducted several experiments over a period of two weeks on a wide area test-bed. Our experiments measure the bandwidth and latency that a overlay provides between the source and the different clients. We also measure the network resource usage and overheads incurred by the different overlay schemes. The details of these measurements are in the sections that will follow. We vary both the source rate and client set to evaluate how well the schemes operate in different conditions.

5.1 Schemes for Constructing Overlays

Our schemes for constructing overlays are derived from the Narada protocol [3], and differ from each other based on which network metrics they consider. We compare the following schemes for overlay construction:

• *Sequential Unicast:* To analyze the efficiency of a scheme for constructing overlays, we would ideally like to compare the overlay tree it produces with the "best possible overlay tree" for the entire set of group members. We approximate this by the Sequential Unicast test, which measures the bandwidth and latency of the unicast path from the source to each recipient *independently* (in the absence of other recipients). Thus, Sequential Unicast is not a feasible overlay at all but a hypothetical construct used for comparison purposes.
• *Random:* This represents a scheme that produces random, but connected overlay trees rooted at the source. This scheme also helps to validate our evaluation, and addresses the issue as to whether our machine set is varied enough that just about any overlay tree yields good performance.
• *Prop-Delay-Only:* This represents a scheme that builds overlays based on propagation delay, a static network metric. Measuring propagation delay incurs low overhead, and overlays optimized for this metric have been shown to yield reasonably good simulation results [3]. In our evaluation, we computed the propagation delay of an overlay link by picking the minimum of several one-way delay estimates.
• *Latency-Only* and *Bandwidth-Only*: These two schemes construct overlays based on a single dynamic metric with no regard to the other metric. They are primarily used to highlight the importance of using both bandwidth and latency in overlay construction.
• *Bandwidth-Latency*: This represents our proposed scheme that uses both bandwidth and latency as metrics to construct overlays.

Many of our hosts are on 10 Mbps connections, and we use source rates as high as 2.4 Mbps. To prevent obviously bad choices of overlay trees due to saturation of the local link, schemes that use static network metrics like *Prop-Delay-Only* are required to impose static, pre-configured degree bound restrictions on the overlay trees they construct [3]. In our evaluation, we try to be give *Random* and *Prop-Delay-Only* the best possible chance to succeed by appropriately choosing per-host degree bounds based on the bandwidth of that host's connection to the Internet. On the other hand, *Bandwidth-Latency*, *Latency-Only* and *Bandwidth-Only* are able to adapt to dynamic network metrics. This enables them to automatically detect and avoid congestion on links near members, without a pre-configured degree bound.

5.2 Experimental Methodology

The varying nature of Internet performance influences the relative results of experiments done at different times. Characteristics may change at any time and affect the performance of various experiments differently. Ideally, we should test all schemes for constructing overlays concurrently, so that they may observe the exact same network conditions. However, this is not possible, as the simultaneously operating overlays would interfere with each other. Therefore, we adopt the following strategy: (i) we interleave experiments with the various protocol schemes that we compare to eliminate biases due to changes that occur at shorter time scales, and (ii) we run the same experiment at different times of the day to eliminate biases due to changes that occur at a longer time scale. We aggregate the results obtained from several runs that have been conducted over a two week period.

Every individual experiment is conducted in the following fashion. Initially, all members join the group at approximately the same time. The source multicasts data at a constant rate and after four minutes, bandwidth and round-trip time measurements are collected. Each experiment lasts for 20 minutes. We adopt the above set-up for all schemes, except *Sequential Unicast*. As described in Section 5.1, *Sequential Unicast* determines the bandwidth and latency information of a unicast path, which we estimate by unicasting data from the source to each receiver for two minutes in sequence.

5.3 Performance Metrics

We use the following metrics to capture the quality of an overlay tree:

• *Bandwidth:* This metric measures the application level throughput at the receiver, and is an indicator of the quality of received video.
• *Latency:* This metric measures the end-to-end delay from the source to the receivers, as seen by the application. It includes the propagation and queuing delays of individual overlay links, as well as queueing delay and processing overhead at end systems along the path. We ideally wish to measure the latency of each individual data packet. However, issues associated with time synchronization of hosts and clock skew adds noise to our measurements of one-way delay that is difficult to quantify. Therefore, we choose to estimate the round trip time (RTT). By RTT, we refer to the time it takes for a packet to move from the source to a recipient along a set of overlay links, and back to the source, using the *same* set of overlay links but in reverse order. Thus, the RTT of an overlay path *S-A-R* is the time taken to traverse *S-A-R-A-S*. The RTT measurements include all delays associated with one way latencies, and are ideally twice the end-to-end delay.
• *Resource Usage*: This metric defined in [3] captures the network resources consumed in the process of delivering data to all receivers. The resource usage of an overlay tree is the sum of the costs of its constituent overlay links. The cost of an overlay link is the sum of the costs of the physical links that constitute the overlay link. In our evaluation, we assume the cost of a physical link to be the propagation delay of that link, guided by the intuition that it is more efficient use of network resources to use shorter links than longer ones. For example, in Figure 1, the cost (delay) of physical link $R1 - R2$ is 25, the cost of the overlay link $A - C$ is 27, and the resource usage of the overlay tree is 31.

We define the *Normalized Resource Usage* of an overlay tree as the ratio of its resource usage to the resource usage with IP Multicast. The resource usage with IP Multicast is the sum of the costs (delays) of the physical links of the native IP Multicast tree used in delivering data to the receivers. In our evaluation, we determine the IP Multicast tree based on the unicast paths from the source to each receiver. This is the tree that the classical DVMRP protocol [4] would construct (assuming Internet routing is symmetrical). We derive the physical links of this IP Multicast tree, as well as the delays of these links, by doing a *traceroute* from the source to each receiver.

Bandwidth and latency are metrics of the application level performance that an overlay provides, while resource usage is a measure of the network costs incurred. The objective of our evaluation is to understand the qualities of the overlay tree that different schemes create with respect to these metrics. For a metric such as resource usage, it is easy to summarize the quality of the overlay produced. However, it is much more difficult to summarize the latency and bandwidth performance that a number of different hosts observe with a few simple metrics. One approach is to present the mean bandwidth and latency, averaged across all receivers. Indeed, we do use this technique in Section 6.1. However, this does not give us an idea of the distribution of performance across different receivers.

A simple approach to summarizing an experiment is to explicitly specify the bandwidth and latencies that each individual receiver sees. Although the set of hosts and source transmission rate are identical, a particular scheme may create a different overlay layout for each experimental run. While an individual host may observe vastly different performance across the runs, this does not imply that the various overlays are of any different quality. Therefore, we need metrics that capture the performance of the overlay tree as a whole.

Let us consider how we summarize an experiment with regard to a particular metric such as bandwidth or latency. For a set of n receivers, we sort the average metric value of the various receivers in ascending order, and assign a *rank* to each receiver from *1* to n. The worst-performing receiver is assigned a rank of *1*, and the best-performing receiver is assigned a rank of n. For every rank r, we gather the results for the receiver with rank r across all experiments, and compute the mean. Note that the receiver corresponding to a rank r could vary from experiment to experiment. For example, the result for rank *1* represents the performance that the worst performing receiver would receive on average in any experiment.

In addition to the mean bandwidth or latency for a given rank, we also calculate the standard deviation of this measure. Variability in performance may occur due to two reasons. First, it arises due to a variation in quality of overlay trees that a particular scheme produces across different runs. For example, a scheme may produce trees where every receiver gets good performance in a particular run, but many receivers get bad performance in another run. Second, variability may also occur due to changes in Internet conditions (such as time of day effects). Thus, potentially no overlay may be able to provide good performance at a given time. However, our results in Section 6 demonstrate that some schemes are able to keep the standard deviation low. This indicates that the standard deviation is a reasonable measure of the variability in performance with a scheme itself.

The metrics above capture the quality of the overlay a scheme constructs. We use a fourth metric, the *Protocol Overhead*, to capture the the overhead incurred by a scheme while constructing overlays. This metric is defined as the ratio of the total bytes of non-data traffic that enters the network to the total bytes of data traffic that enters the network. The overhead includes control traffic required to keep the overlay connected, and the probe traffic and active bandwidth measurements involved in the self-organization process.

5.4 Implementation Issues

The experiments are conducted using unoptimized code running at the user level. Implementation overhead and delays at end systems could potentially be minimized by pushing parts of the implementation in the kernel, and by optimizing the code. We have used TFRC [5] as the underlying transport protocol on each overlay link, as discussed in Section 3. TFRC is rate-controlled UDP, and achieves TCP-friendly bandwidths. It does not suffer delays associated with TCP such as retransmission delays, and queueing delays at the sender buffer.

6. EXPERIMENTAL RESULTS

We begin by presenting results in a typical experiment run in Section 6.1. Section 6.2 provides a detailed comparison of various schemes for constructing overlays with regard to application level performance, and Section 6.3 presents results related to network costs.

6.1 Results with a Typical Run

The results in this section give us an idea of the dynamic nature of overlay construction, and how the quality of the overlay varies with time. Our experiment was conducted on a week-day afternoon, using the *Primary Set* of machines and at a source rate of 1.2 Mbps. The source host is at UCSB.

Figure 3 plots the mean bandwidth seen by a receiver, averaged across all receivers, as a function of time. Each vertical line denotes a change in the overlay tree for the source UCSB. We observe that it takes about 150 seconds for the overlay to improve, and for the hosts to start receiving good bandwidth. After about 150 seconds, and for most of the session from this time on, the mean bandwidth observed by a receiver is practically the source rate. This indicates that all receivers get nearly the full source rate throughout the session.

Figure 4 plots the mean RTT to a receiver, averaged across all receivers as a function of time. The mean RTT is about 100 ms after about 150 seconds, and remains lower than this value almost throughout the session.

Figures 3 and 4 show that in the first few minutes of the session, the overlay makes many topology changes at very frequent intervals. As members gather more network information, the quality of the overlay improves over time, and there are fewer topology changes. In most of our runs, we find that the overlay converges to a reasonably stable structure after about four minutes. Given this, we gather bandwidth and RTT statistics after four minutes for the rest of our experiments. We present ideas for how convergence time may be minimized in the future in Section 8.

The figures above also highlight the adaptive nature of our scheme. We note that there is a visible dip in bandwidth, and a sharp peak in RTT at around 460 seconds. An analysis of our logs indicates that this was because of congestion on a link in the overlay tree. The overlay is able to adapt

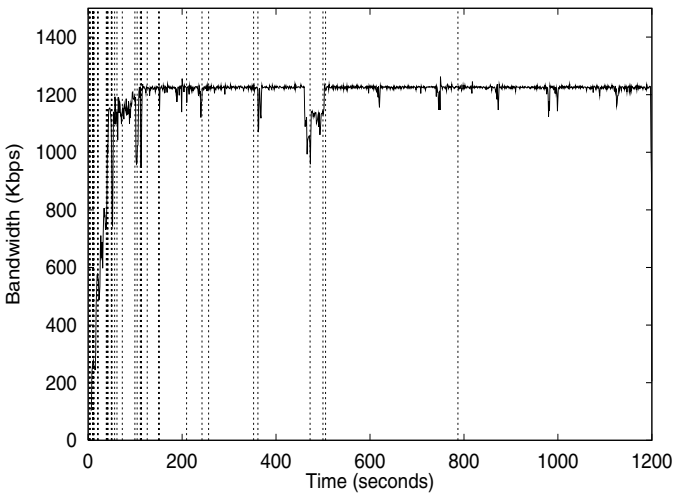

Figure 3: Mean Bandwidth averaged over all receivers as a function of time.

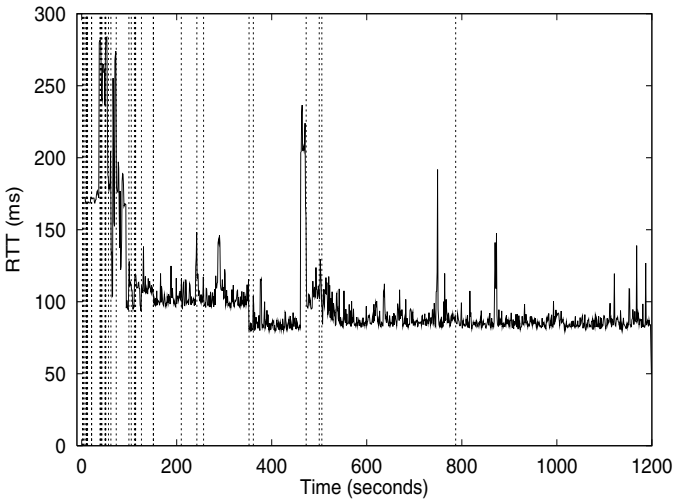

Figure 4: Mean RTT averaged over all receivers as a function of time.

by making a set of topology changes, as indicated by the vertical lines immediately following the dip, and recovers in about 40 seconds.

We now consider how the RTTs to individual receivers vary during a session. Figure 5 plots the cumulative distribution of the RTT estimates to every receiver. For each receiver, there is usually at least one RTT estimate every second, for the entire duration of the session. Each curve corresponds to a particular receiver, and each point indicates the fraction of the RTT estimates to that receiver that have an RTT lower than a particular value. For all receivers, over 94% of the RTT estimates are less than 200 ms, while over 98% of the RTT estimates are less than 400 ms. Assuming that one-way latency is one half of the RTT, this indicates that end-to-end latencies are lower than 100 ms most of the time, and less than 200 ms almost all the time.

6.2 Comparison of Schemes for Overlays

We present detailed results of our comparisons of several schemes for constructing overlay trees on the Internet. We

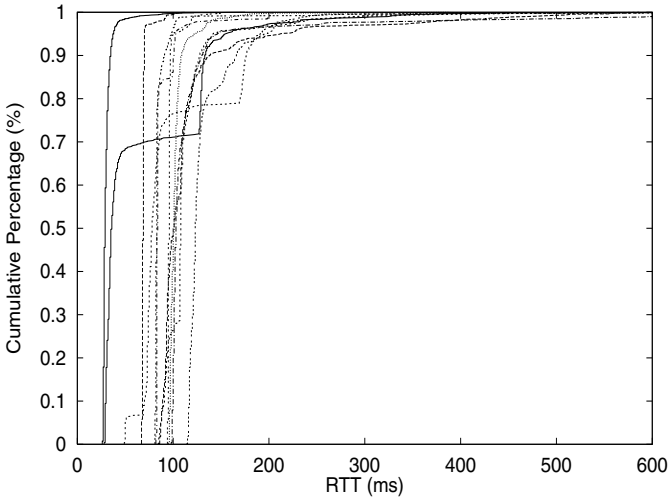

Figure 5: Cumulative distribution of RTT, one curve for each receiver.

begin our comparison study with the *Primary Set* and a source rate of 1.2 Mbps. Internet paths between most pairs of hosts in the *Primary Set* can sustain throughputs of 1.2 Mbps. Thus, this study represents a relatively less heterogeneous environment where simpler schemes could potentially work reasonably well. Next, we consider the *Primary Set*, but at a source rate of 2.4 Mbps. This environment is more stressful to our schemes for two reasons. First, fewer Internet paths in the *Primary Set* are able to sustain this increased source rate and thus, this represents an environment with a higher degree of heterogeneity. Second, several hosts in our test-bed are located behind 10 Mbps connections, and a poorly constructed overlay can result in congestion near the host. Thus, schemes that work well at 1.2 Mbps potentially work less well at 2.4 Mbps. Finally, to stress our scheme *Bandwidth-Latency*, we consider an extremely heterogeneous environment represented by the *Extended Set*, and assuming a source rate of 2.4 Mbps. We believe our choice of source rates is realistic and representative of current and emerging high bandwidth video applications.

6.2.1 Primary Set at 1.2 Mbps Source Rate

Figure 6 plots the mean bandwidth against rank for four different schemes. Each curve corresponds to one scheme, and each point in the curve corresponds to the mean bandwidth that a machine of that rank receives with a particular scheme, averaged across all runs. The error-bars show the standard deviation. Thus they do not indicate confidence in the mean, rather they imply the degree of variability in performance that a particular scheme for constructing overlays may involve. For example, the worst-performing machine (rank 1) with the *Random* scheme, receives a bandwidth of a little lower than 600 Kbps on average. We use the same way of presenting data in all our comparison results.[1]

We wish to make several observations. First, the *Sequential Unicast* test indicates that all but one machine get close to the source rate, as indicated by one of the top lines with a dip at rank 1. Second, both *Bandwidth-Latency* and *Bandwidth-Only* are comparable to *Sequential Unicast*. They are able to ensure that even the worst-performing ma-

[1]The curves are slightly offset from each other for clarity of presentation.

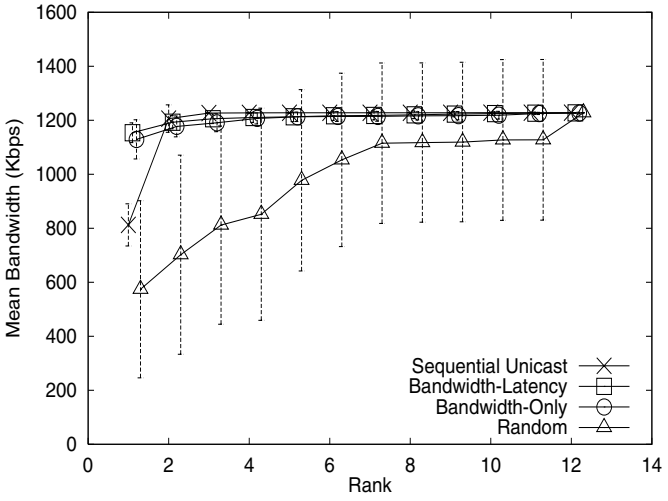

Figure 6: Mean bandwidth versus rank at 1.2 Mbps source rate for the *Primary Set* of machines

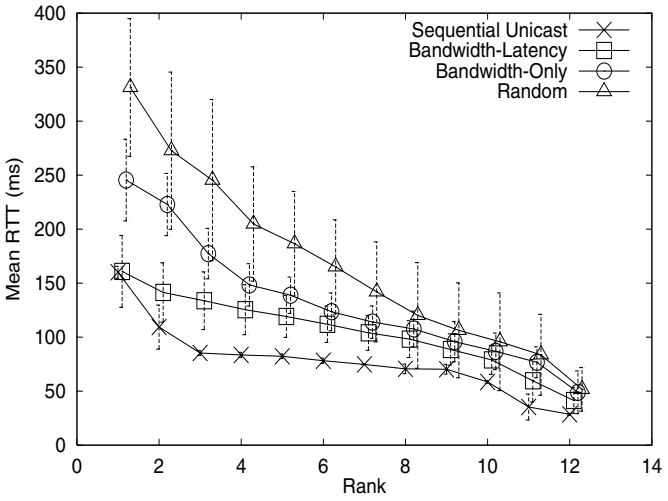

Figure 7: Mean RTT versus rank at 1.2 Mbps source rate for the *Primary Set* of machines

chine in any run receives 1150 Kbps on average. Further, these schemes can result in consistently good performance, as indicated by the small standard deviations. Interestingly, these schemes result in much better performance for the worst-performing machine as compared to *Sequential Unicast*. It turns out this is because of the existence of pathologies in Internet routing. It has been observed that Internet routing is sub-optimal and there often exists alternate paths between end system that have better bandwidth and latency properties than the default paths [12]. Third, the *Random* scheme is sub-optimal in bandwidth. On average, the worst-performing machine with the *Random* scheme (rank 1) gets a mean bandwidth of about 600 Kbps. Further, the performance of *Random* can be quite variable as indicated by the large standard deviation. We believe that this poor performance with *Random* is because of the inherent variability in Internet path characteristics, even in relatively well connected settings.

Figure 7 plots mean RTT against rank for the same set

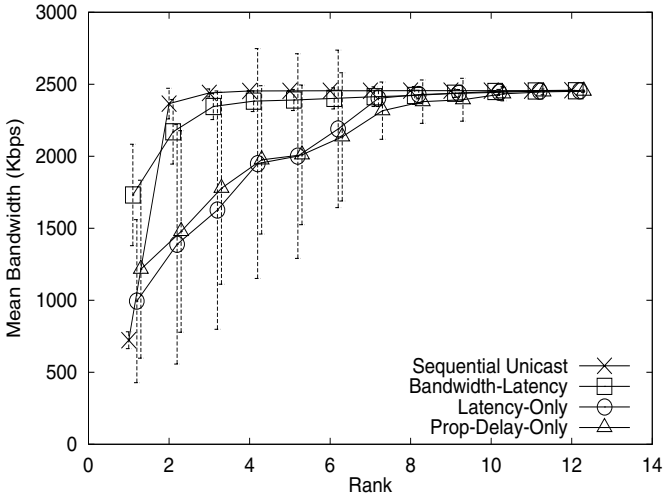

Figure 8: Mean bandwidth versus rank at 2.4 Mbps source rate for the *Primary Set*

of experiments. First, the RTT of the unicast paths from the source to the recipients can be up to about 150 ms, as indicated by the lowest line corresponding to *Sequential Unicast*. Second, *Bandwidth-Latency* is good at optimizing the overlay for delay. The worst machine in any run has an RTT of about 160 ms on average. Third, both *Random* and *Bandwidth-Only* perform considerably worse. While *Random* results in an RTT of about 350 ms for the worst machine on average, *Bandwidth-Only* results in an RTT of about 250 ms. Both *Bandwidth-Only* and *Random* can have poor latencies because of suboptimal overlay topologies that may involve criss-crossing the continent. In addition, *Random* is unable to avoid delays related to congestion, particularly near the participating end hosts, while *Bandwidth-Only* may benefit due to some correlation between bandwidth and delay.

We have also evaluated *Prop-Delay-Only* and *Latency-Only* under this setting, and find that they performs similarly to *Bandwidth-Latency* in RTT, and slightly worse in bandwidth. We omit the results for clarity. Further, given the poor performance of *Random*, even in very simple settings, we do not consider it further in our evaluation.

6.2.2 Primary Set at 2.4 Mbps Source Rate

In this section, we focus on the performance comparison between *Bandwidth-Latency* and two delay-based schemes, *Prop-Delay-Only* and *Latency-Only*. Figures 8 and 9 present the mean bandwidth and RTT against host rank for four different schemes.

First, we observe that the paths from the source to most receivers can sustain bandwidths of up to 2.4 Mbps, as indicated by the *Sequential-Unicast* test. Second, *Bandwidth-Latency* comes very close to achieving this benchmark, and can outperform *Sequential Unicast* for machines with lower rank. Next, we observe that both *Latency-Only* and *Prop-Delay-Only* perform poorly in bandwidth. For machines of rank 1–5, *Bandwidth-Latency* can outperform *Prop-Delay-Only* and *Latency-Only* by over 500 Kbps. While *Prop-Delay-Only* and *Latency-Only* can provide reasonable performance at source rates of 1.2 Mbps, they are unable to provide good performance in bandwidth at 2.4 Mbps with the same set of machines.

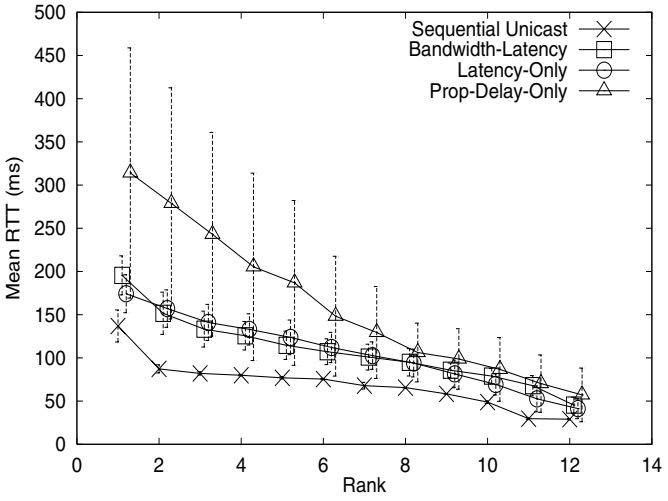

Figure 9: Mean RTT versus rank at 2.4 Mbps source rate for the *Primary Set*

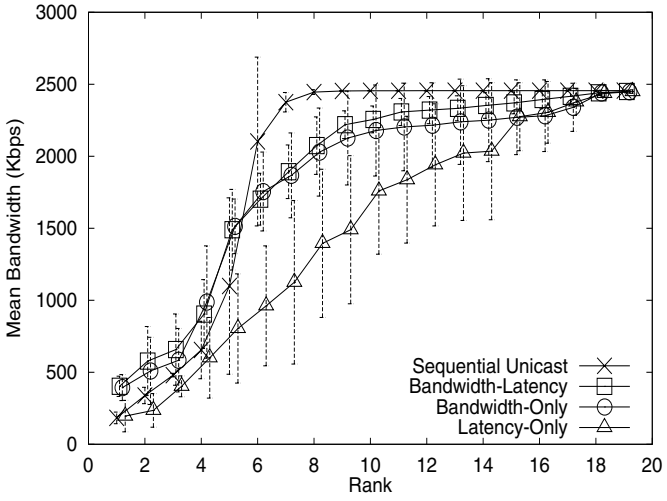

Figure 10: Mean bandwidth versus rank at 2.4 Mbps source rate for the *Extended Set* of machines

From the perspective of RTT, we find that *Bandwidth-Latency* performs almost indistinguishably from *Latency-Only*, and both schemes achieve performance reasonably close to *Sequential Unicast*. However, more surprisingly, *Prop-Delay-Only* achieves RTTs at least 100 ms more than *Bandwidth-Latency* for machines of lower rank, and thus performs badly even in the RTT metric. This is because delays in the Internet may often arise due to congestion, and optimizing purely for propagation delay need not optimize the latencies seen by the application. This observation becomes particularly important in our environment where many hosts are behind 10 Mbps connections, and poorly constructed overlays could cause congestion near a host. While we did use conservative pre-configured degree bounds recommended in [2, 3, 6], this strategy is not capable of dealing with dynamic cross-traffic. In contrast, the dynamic nature of *Bandwidth-Latency* and *Latency-Only* enables them to perform better in such situations.

We have also evaluated Bandwidth-Only in this environment. We find the bandwidth results are comparable to *Bandwidth-Latency*, but the RTT results are worse. Finally, because of the poor performance of *Prop-Delay-Only*, our future evaluation concentrates on *Latency-Only* while analyzing the performance of delay based schemes.

6.2.3 Extended Set at 2.4 Mbps Source Rate

Our results so far demonstrate that even in less heterogeneous environments such as the *Primary Set*, good performance requires considering both bandwidth and latency as metrics in overlay construction. To further emphasize the importance of taking both bandwidth and latency into account, we consider extremely heterogeneous environments as represented by the *Extended Set*. Figures 10 and 11 plot the bandwidth and RTT against host ranks for the four schemes of interest.

The *Sequential Unicast* curves show that there are quite a few members that have low bandwidth and high latencies from the source, which indicates the heterogeneity in the set we consider. Even in such a heterogeneous setting, *Bandwidth-Latency* is able to achieve a performance close to the *Sequential Unicast* test. Apart from the less well-connected hosts (ranks 1–5), all other members get bandwidths of at least 1.8 Mbps, and see RTTs of less than 250 ms on average. For ranks 1–5, *Bandwidth-Latency* is able to exploit Internet routing pathologies and provide better performance than *Sequential Unicast* . A particularly striking example was two machines in Taiwan, only one of which had good performance to machines in North America. In our runs, the machine with poorer performance was able to achieve significantly better performance by connecting to the other machine in Taiwan.

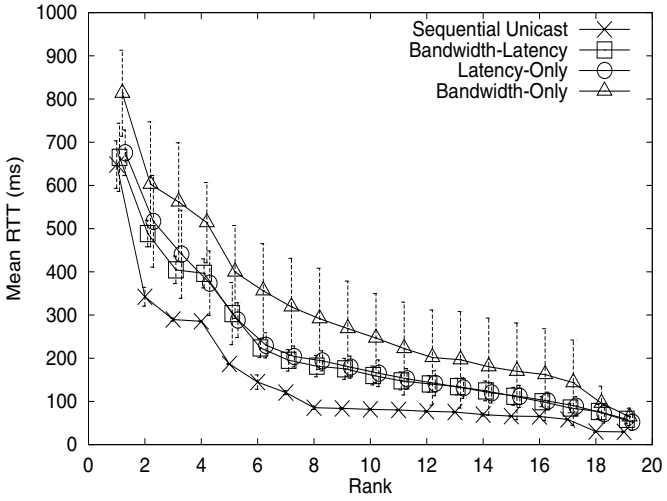

Figure 11: Mean RTT versus rank at 2.4 Mbps source rate for the *Extended Set* of machines

Next, we observe that *Bandwidth-Only* results in high RTT, while *Latency-Only* performs poorly in bandwidth. For example, for machines of rank 7, *Bandwidth-Latency* can sustain throughputs almost 800 Kbps more than *Latency-Only*, and can achieve RTTs more than 100 ms lower than *Bandwidth-Only*. Further, *Bandwidth-Latency* has smaller standard deviations, which indicates that the overlays it produces consistently attain good performance in both bandwidth and latency. Finally, we observe that *Bandwidth-Latency* provides equivalent performance to *Bandwidth-Only* in bandwidth, and to *Latency-Only* in RTT indicating that

Experiment Setup	Primary 1.2 Mbps	Primary 2.4 Mbps	Extended 2.4 Mbps
Unicast	2.62	2.62	1.83
Random	2.24	2.05	1.97
Latency-Only	1.39	1.42	1.25
Bandwidth-Only	1.85	1.86	1.51
Bandwidth-Latency	1.49	1.73	1.31
Min-Span	0.85	0.85	0.83

Table 1: Average normalized resource usage of different schemes

optimizing for multiple metrics does not compromise the performance with respect to any single metric.

6.2.4 Summary of comparison results

We summarize results from our comparison study below:

• Our techniques enable the construction of efficient overlays optimized for both bandwidth and latency, as required by conferencing applications. Even in extremely heterogeneous environments, *Bandwidth-Latency* has performance comparable to *Sequential Unicast*, and sometimes performs better by exploiting Internet pathologies.
• Random overlays do not perform well even in settings with a small amount of heterogeneity.
• Overlays that adapt to simple static metrics like propagation delay perform quite poorly, not only in bandwidth, but also in latencies. This is because schemes that use only static metrics fail to detect and react to network congestion, resulting in larger queueing delays and higher packet loss.
• Overlays that adapt to latency alone are unable to provide good bandwidth performance, especially at higher source rates. Conversely, overlays that adapt to bandwidth alone are unable to provide good latencies. These results indicate that latency and bandwidth are not strongly correlated on the Internet, and it is critical to consider both metrics to construct overlays optimized for conferencing applications.

6.3 Network Level Metrics

Table 1 compares the mean normalized resource usage (Section 5.3) of the overlay trees produced by the various schemes for different environments and source rates. The values are normalized with respect to the resource usage with native IP Multicast support, determined as explained in Section 5.3. Thus, we would like the normalized resource usage to be as small as possible, with a value of 1.00 representing an overlay tree that has the same resource usage as IP Multicast. Given that the trees constructed by self-organizing protocols can change over time, we consider the final tree produced at the end of an experiment. However, we observe that the overlays produced by these schemes are reasonably stable after about four minutes.

There are several observations to be made from Table 1. First, naive degenerated unicast trees which have all recipients rooted at the source, and schemes such as *Random* that do not explicitly exploit network information have a high resource usage. Second, protocols that adapt to bandwidth alone (*Bandwidth-Only*) make less efficient use of network resources compared to protocols such as *Bandwidth-Latency*, and *Latency-Only* which consider delay based metrics. Third, the resource usage with *Bandwidth-Latency* is a little higher than *Latency-Only*, which reflects the cost in adapting to better bandwidth paths. Finally, the resource usage with *Bandwidth-Latency* increases with source rate (in the *Primary Set*). This is because it favors higher bandwidth paths over lower delay paths at higher source rates.

Experiment Setup		Primary 1.2 Mbps	Primary 2.4 Mbps	Extended 2.4 Mbps
Average Overhead (%)		10.79	11.53	14.20
% of overhead due to	Bandwidth Probes	92.24	96.03	94.30
	Other	7.76	3.37	5.70

Table 2: Average overhead with *Bandwidth-Latency* and a breakdown of the overhead

We have also determined the resource usage of *Min-Span*, the minimum spanning tree of the complete graph of all members, computed by estimating the delays of all links of the complete graph. Minimum spanning trees are known to be optimal with respect to resource usage, and as Table 1 shows, have lower resource usage than IP Multicast. This indicates that an End System Multicast architecture can indeed make as efficient, if not better use of network resources than IP Multicast. However, while minimum spanning trees are efficient from the network perspective, it is not clear that they are efficient from the application perspective.

Table 2 summarizes the protocol overhead (Section 5.3) involved in *Bandwidth-Latency*, along with a breakdown of the main factors that contribute to the overhead. We find that the average overhead is between 10 to 15% across all settings. This is an indication that even simple heuristics that we have implemented can keep the overall overhead low. Further, more than 90% of the overhead is due to members probing each other for bandwidth. Other sources of overhead contribute just 3–7% of the overhead. These include exchange of routing messages between neighbors, group management algorithms to keep the overlay connected, and probes to determine the delay and routing state of remote members. This confirms our intuition that techniques for lowering protocol overhead must focus on reducing costs of bandwidth probes.

An analysis of our logs reveals that the simple heuristics introduced in Section 4.2 are able to reduce around 50% of possible bandwidth probes between pairs of members over a 20 minute period. Further, in experiments with the *Extended Set*, we find that there are no bandwidth probes to the member behind the ADSL connections. Further, there are very few probes from a machine in North America to hosts in Asia and Europe. While the results are encouraging, overhead due to active bandwidth measurements can be a concern as one considers larger sized groups. We present some of our ideas for tackling this in Section 8.

7. ADAPTING TO NETWORK DYNAMICS

Section 6 has shown that it is important to adapt to dynamic metrics such as bandwidth and latency while constructing overlays optimized for conferencing applications. When network congestion occurs, overlays need to adapt by making appropriate topology changes in a timely fashion to ensure good and stable application performance. In this section, we evaluate how quickly our protocol can adapt to changes in network conditions, and study the factors that contribute to adaptation times.

To study these issues, we design a set of experiments on the Internet where we artificially introduce network congestion on selected links in the overlay tree, and evaluate how our system responds to these congestion signals. Our study is conducted using our *Bandwidth-Latency* overlay scheme.

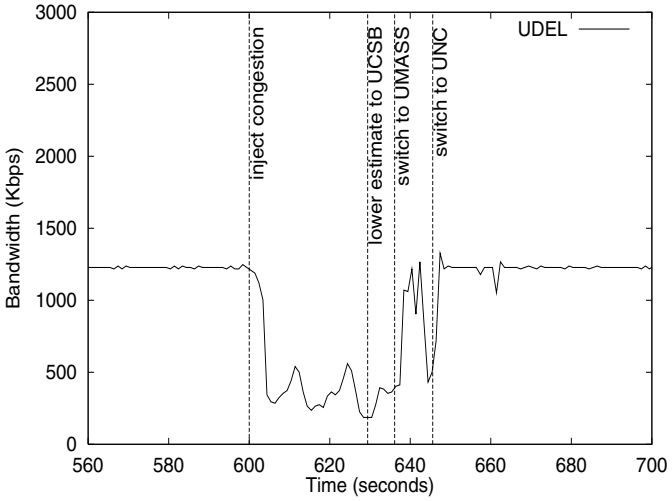

Figure 12: An example demonstrating adaptation to network congestion in one controlled experiment.

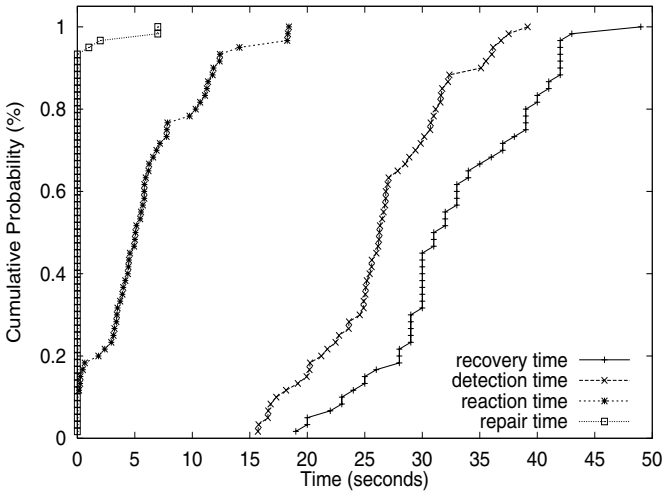

Figure 13: Cumulative distribution of recovery time and its components for the victim hosts

All experiments are conducted using the hosts in the *Primary Set*. The source, located at UCSB, sends CBR traffic at 1.2 Mbps. Ten minutes into the experiment, we randomly choose a *victim* from among the receivers, and simulate a state of congestion on the link between the victim and its parent in the rest of the experiment. We simulate congestion by randomly dropping 10% of the packets at the parent. We do not model increased latencies that are typical during network congestion. Given that each link in the overlay tree is congestion-controlled using TFRC, a high loss rate results in a substantial decrease in the bandwidth performance of that link. As a result, the bandwidth received by the victim drops sharply. Moreover, if the victim is not a leaf in the overlay tree, all the *descendants* of the victim will suffer the same performance degradation in bandwidth. Given that we are not modeling latency effects of congestion, our experiments evaluate our techniques in adapting to bandwidth changes alone.

To explain the issues involved in the adaptation process, we present the behavior of the victim in a particular experiment. Figure 12 plots the bandwidth the victim gets as a function of time. Each vertical line indicates an event of interest. In this experiment, a machine at U. Delaware (UDEL) is the victim. Before congestion is introduced, UDEL receives data directly from the source UCSB. At time 600, we introduce congestion on the link between UCSB and UDEL, and UDEL observes a substantial drop in bandwidth. After switching parents twice, UDEL starts observing good performance 43 seconds later. Formally, we consider a receiver to have *recovered* from congestion when it continuously receives more than 90% of the *stable bandwidth* for 15 seconds. We determine the stable bandwidth of a host based on the bandwidth it receives right before congestion is introduced. We define *recovery time* as the time it takes for a host to recover, since the onset of congestion.

To gain further insight in the adaptation process of the protocol, we divide the recovery time into three components as follows:

• *Detection time:* This is the duration of time for which congestion must persist before a victim recognizes this as an event it should adapt to. In our scheme, we consider congestion to have been recognized when the advertised bandwidth of a link moves down by one discretized level (Section 4.1). In this experiment, the detection time is 30 seconds.

• *Reaction time:* This is the amount of time the protocol takes to make the first parent change after congestion is detected. In this experiment, UDEL switches to the host UMASS in 6 seconds.

• *Repair time:* This is the additional amount of time the protocol takes to fully recover since the first parent change. In this experiment, UDEL switches to the host UNC 7 seconds after switching to UMASS.

The detection time captures the time scale at which an overlay has been designed to adapt to congestion. The reaction and repair times however measure how quickly our schemes react given that a decision to adapt has been made.

We now present summary statistics from 60 controlled experiments. The victims have no descendants in 24 experiments, and an average of 3.1 descendants in the other experiments.

First we analyze the behavior of the victim hosts alone. Figure 13 shows the cumulative distribution of each metric we consider. For most of the cases, recovery time is between 20 and 45 seconds. Further, the largest contributory factor is detection time, which is usually between 20 and 35 seconds. In contrast, reaction time and repair time are relatively short. Narada is a mesh-based protocol running a distance vector algorithm on the mesh, and two different mechanisms can contribute to reaction time in Narada. In most of the cases, the victim chooses another neighbor in the mesh as its new parent. Thus the reaction time in these cases depends on the frequency of route exchanges between neighbors, which currently occur once every ten seconds. In a few cases however, none of the mesh neighbors may offer a better path to the source, perhaps because all other neighbors are children of the victim. In such cases the victim must add a new mesh link to improve performance, and this process can take longer. Finally, we observe that the repair time is zero for more than 90% of the cases. This indicates that typically only a single parent change is involved in the adaptation process.

Next we analyze the recovery time of the descendant hosts,

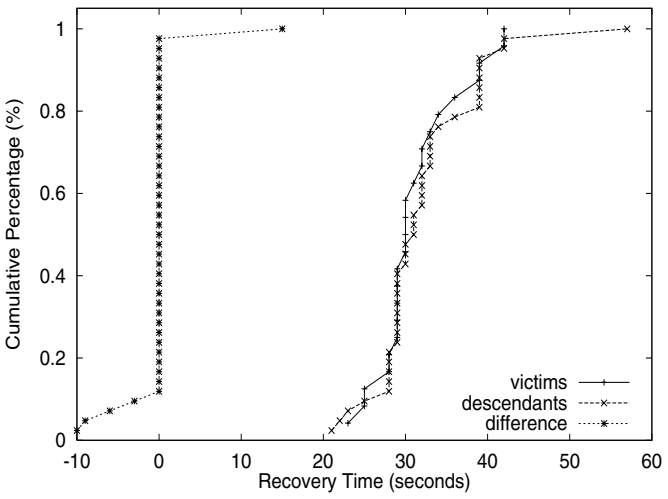

Figure 14: Cumulative recovery time of victims and their descendants

considering only runs where victims have descendants. We define *difference* as the difference in recovery time between the descendant and its respective victim in the same experiment. A positive difference implies that the descendant recovers slower than its victim. Figure 14 shows the cumulative distribution of the recovery time of the descendants and the victims, as well as the difference. We observe that the recovery time distribution for victims and descendants exhibit similar trends. Moreover, more than 80% of descendants recover at the same time as their respective victims, as shown by the *difference* curve. Further analysis indicates that about 75% of the descendants recover without a change of parent. In general, it is desirable to minimize the number of overlay changes when adapting to network congestion, as it increase the stability of the overlay structure.

Our results indicate that the detection time is a significant fraction of the recovery time. Further, the detection times are around 20 to 35 seconds. Our design choice has been motivated by two conflicting concerns. If the overlay adapts very quickly to congestion that is usually short in duration, the benefit of adaptation is small and the overlay could potentially become unstable. However, if the overlay adapts too slowly, the applications would suffer a performance penalty during the transient period. Although we have found our choice of detection time to work reasonably well in practice, we believe further studies are needed to determine the proper time scale of adaptation, grounded on a more rigorous understanding of the characteristics of Internet congestion.

8. DISCUSSION

Our results indicate that End System Multicast can meet the stringent bandwidth and latency demands of conferencing applications in heterogeneous and dynamic Internet environments. Further, to achieve good performance, it is important that overlays dynamically adapt to both bandwidth and latency.

This paper exposes three issues that we hope to explore in the future. First, to construct overlays optimized for conferencing, we employ active end-to-end measurements to estimate path bandwidths. While active measurements can be costly, simple techniques employed in this paper help to restrict the overhead to about 10–15% for groups as large as twenty members. However, it is unclear whether the overhead results scale to larger group sizes. Second, in the absence of initial network information, self-organizing protocols must take some time to discover network characteristics and converge to efficient overlays. Currently the convergence time is about 4 minutes. While this may be acceptable in conferencing applications which typically have a long duration, it may become an issue in other real-time applications. Finally, our current protocol is designed to adapt to network congestion on the time scale of tens of seconds. This design choice is motivated by the inherent tradeoff between the time scale of adaptation and the stability of overlay topology. While adaptation at such time scales may be acceptable when operating in less dynamic environments, transient degradation of application performance may become an important issue in highly dynamic environments.

As mentioned in Section 2, End System Multicast occurs in two distinct architectural flavors: peer-to-peer architectures, and proxy based architectures. The discussion in this paper has centered on End System Multicast in general, rather than the specific architectural instantiation. However, we believe that there are opportunities to address these new issues in an architecture-specific way.

Proxy based architectures can have several advantages. First, multiple multicast groups may share the same proxy service, which enables sharing of network performance information across groups. Second, proxies are persistent beyond the lifetime of individual groups. This allows proxies to exploit past history of network performance. Sharing of network information, and leveraging past history help to reduce the number of active measurements needed in constructing overlays. Further, efficient overlays may be quickly created when new groups are instantiated. A third advantage of proxy architectures is that they are better provisioned. We expect that proxy environments have more stable network performance and can leverage the QoS infrastructure on the network more easily. Finally, proxies are the natural coordination points for managing (network) resources among multiple groups that use the proxy service. Thus, when congestion occurs, proxies can allocate more resources to groups with more stringent performance requirements.

While proxy based architectures can have several advantages, peer-to-peer architectures have the attractive property that they are completely distributed and can scale to support millions of groups. This is because each end host keeps state only for the small number of groups in which it actually participates. We are currently investigating a multi-path data delivery framework targeted at these architectures, where each recipient gets (potentially redundant) data from the source along multiple paths, with a fraction of the data flowing along any given path. This multi-path framework has the potential to address the new issues raised in this paper. First, because data is received along multiple paths, degradation in performance along any individual path does not radically affect the overall performance seen by the recipient. This enables robust application level performance even over shorter time scales. Further, data delivery can be made even more robust with the use of redundancy and error correction. Second, the multi-path framework enables a recipient to monitor the performance of several overlay links concurrently, including (potentially poor) links for which no previous bandwidth estimate is available. This in turn minimizes the need for active bandwidth measurements, and can

lead to improved performance during the time it takes the overlay to converge to a good structure.

9. RELATED WORK

Several studies [2, 3, 9, 11] present detailed simulation results to argue that the performance penalties associated with an overlay based approach may be low. To our knowledge, this is the first detailed evaluation of self-organizing overlay protocols in a dynamic and heterogeneous Internet environment.

Several self-organizing protocols have been proposed recently in an overlay setting. Gossamer [2], Narada [3], Yoid [6] and most recently, Bayeux [13], have only considered delay based metrics. Further, they have not addressed important issues pertaining to the dynamic nature of these metrics. Overcast [9] is targeted at broadcasting and reliable data-delivery applications where delay is not a concern. Thus, it constructs overlay trees optimized for bandwidth alone. However, as our results suggest, considering both bandwidth and delay based metrics could potentially enable more efficient use of network resources without sacrificing application performance.

ALMI [11] has advocated a centralized approach to overlay multicast where algorithms for group management and overlay optimization are coordinated at a central point. ALMI currently optimizes the overlay for network resource usage by constructing minimum spanning trees. However, while minimum spanning trees are efficient from the network perspective, it is not clear that they can perform well from the application perspective.

10. SUMMARY

Our results indicate that End System Multicast is a viable architecture for enabling performance demanding audio and video conferencing applications in dynamic and heterogeneous Internet settings. In experiments with our *Primary Set*, at source rates of both 1.2 and 2.4 Mbps, most hosts are able to sustain over 95% of the source rate on average, and yet achieve latencies of less than 100 ms. In extremely heterogeneous settings such as the *Extended Set*, the mean performance attained by each receiver is comparable to the performance of the unicast path from the source to that receiver.

Our results indicate that to achieve good performance for conferencing applications, it is critical to consider both bandwidth and latency while constructing overlays. For example, in experiments with the *Extended Set*, *Bandwidth-Latency* can provide 50% higher throughput than *Latency-Only*, and 30–40% lower latencies than *Bandwidth-Only* for several ranks. Protocols that do not consider any network metrics like *Random*, or those that consider only static network metrics like *Prop-Delay-Only* perform much worse.

Our techniques introduce a network overhead of 10–15% for groups with 20 members. Further, they are designed to adapt at the time scale of 20–35 seconds. Adaptation at this time scale may be sufficient in less dynamic environments, and in applications which are willing to tolerate occasional glitches in performance. Indeed, our techniques worked reasonably well in the realistic Internet environments we consider. Our future work includes exploring mechanisms for achieving shorter time scale adaptation targeted at extremely dynamic environments, and mechanisms for lowering network costs for larger sized groups

ACKNOWLEDGEMENTS

We are deeply grateful to our contacts at over twenty institutions who gave us guest accounts and tolerated our experiments. Tung Fai Chan and Annie Cheng developed a cool visualization tool that helped us develop and debug Narada. We thank Jason Flinn, Jun Gao, Jorjeta Jetcheva and the anonymous referees for comments that helped improve the presentation of the paper.

11. REFERENCES

[1] Deepak Bansal and Hari Balakrishnan. Binomial Congestion Control Algorithms. In *Proc. IEEE INFOCOM*, April 2001.

[2] Y. Chawathe. Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service. Fall 2000. Ph.D. thesis.

[3] Y. Chu, S. Rao, and H. Zhang. A Case for End System Multicast. In *Proceedings of ACM Sigmetrics*, June 2000.

[4] S. Deering. Multicast Routing in Internetworks and Extended Lans. In *Proceedings of ACM SIGCOMM*, August 1988.

[5] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-Based Congestion Control for Unicast Applications. In *Proceedings of ACM SIGCOMM*, August 2000.

[6] P. Francis. Yoid: Your Own Internet Distribution, http://www.aciri.org/yoid/. April 2000.

[7] V. Hardman, A. Sasse, M. Handley, and A. Watson. Reliable audio for use over the Internet. In *Proceedings of INET*, June 1995.

[8] V. Jacobson and S. McCanne. Visual audio tool (vat). http://www-nrg.ee.lbl.gov/vat/

[9] J. Jannotti, D. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole Jr. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the Fourth Symposium on Operating System Design and Implementation (OSDI)*, October 2000.

[10] S. McCanne and V. Jacobson. vic: A flexible framework for packet video. In *ACM Multimedia*, November 1995.

[11] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. ALMI: An Application Level Multicast Infrastructure. In *Proceedings of the 3rd Usenix Symposium on Internet Technologies & Systems (USITS)*, March 2001.

[12] S. Savage, A. Collins, E. Hoffman, J.Snell, and T. Anderson. The end-to-end effects of Internet path selection. In *Proceedings of ACM Sigcomm*, August 1999.

[13] S.Q.Zhuang, B.Y.Zhao, A.D.Joseph, R.H.Katz, and J.D.Kubiatowicz. Bayeux: An architecture for scalable and fault-tolerant wide-area data dissemination. In *Proceedings of NOSSDAV*, 2001.

[14] W. Tan and A. Zakhor. Real-time Internet video using error resilient scalable compression and tcp-friendly transport protocol. In *IEEE Trans. Multimedia, Vol. 1, No. 2*, June 1999.

[15] Z. Wang and J. Crowcroft. Bandwidth-delay based routing algorithms. In *IEEE GlobeCom*, November 1995.