Establishing the viability of End System Multicast using a Systems Approach to Protocol Design

Sanjay G. Rao

CMU-CS-04-168
October 16, 2004

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA

Thesis Committee:
Hui Zhang, Chair
Christos Faloutsos
Bruce M. Maggs
Srini Seshan
Jim Kurose, University of Massachussetts, Amherst

Submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy

Copyright © 2004 by Sanjay G. Rao

This research was sponsored by the National Science Foundation (NSF) under grant nos. ANI-0326472, ANI-9814929, ANI-0085920, ANI-0331653, NCR-9624979, ANI-9730105 and by the US Air Force Research Laboratory (AFRL) under grant no. F306029720287. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the NSF, AFRL, the U.S. government or any other entity.

Keywords: Group Communication, Broadcasting, Conferencing, Multicast, Overlays, Self-organization, Distributed Protocols, Peer-to-Peer

Acknowledgements

I entered the Ph.D program with mixed feelings about the relevance of academic research. Working with Hui has been an exciting experience that not only helped eliminate my skepticism, but has motivated me to an extent that I have decided to pursue an academic career. I view my Ph.D experience as being a privileged opportunity to be an apprentice to one of the best masters in the craft, and it is impossible to summarize the number of things I have learnt in one short paragraph.

Much of this work has been done in close collaboration with Yang-hua who has been a selfless team-player, and a close friend. I treasure all our animated discussions on ESM, involving heated arguments and disagreements on practically every issue. Eugene, Kay, Yang-hua and I grew up doing our Ph.Ds together, and have several pleasant memories to share. Special thanks to Jibin and Aditya, and the rest of the ESM team for their contributions to the ESM project, which have directly affected the success of this thesis.

A lot of this work in this thesis has benefited from detailed technical conversations and discussions with Srini Seshan. The last chapter in this thesis has been done in collaboration with Ashwin Bharambe, Srini and Venkat Padmanabhan. Jim Kurose provided detailed comments on drafts of the thesis, and has always taken time out of his tight schedule to provide feedback and advice. I thank Bruce Maggs and Christos Faloutsos for their support, encouragement and feedback.

Jason, Dushyanth, David, George, Jeff, Umair, Jun Gao and many others have made Wean Hall a habitable place. Karthik, Sita, Raj, Vikram, and several other friends helped make my life outside of Wean Hall some of the best years in my life. I would like to particularly thank Kumar, whose Ph.D lifetime overlapped with mine, and with whom I had millions of conversations sharing our aspirations, frustrations, and the ups and downs of life. I thank both Kumar and Archana for several excellent dinners over the years.

Completing a Ph.D and going through an academic job-search process is a stressful experience. I would like to thank my wife Ranjani for her love, support, patience, understanding, positive spirit, and innumerable sacrifices over the last few years, which made this possible. My parents have provided rock-like support throughout my years at Carnegie Mellon, and their blessings, prayers, and wisdom have been a great source of strength. I would also like to thank my extended family for their wishes. In particular my uncle Dr. Krishnamurthy Rao has been a source of inspiration, and took a strong personal interest in my progress.

Abstract

This thesis is motivated by the vision of enabling the ubiquitous deployment of applications such as audio/video conferencing and broadcasting over the Internet. For over 15 years, researchers have attempted to enable such applications using the IP Multicast architecture. However, concerns regarding per-group state in routers, deployment issues, and difficulties with supporting higher level functionality such as reliability and congestion control has prevented IP Multicast from taking roots.

We take the stand that *"it is feasible to efficiently enable group communication applications on the Internet without router and IP level support"* . We demonstrate this thesis in the context of an alternate architecture that we call *End System Multicast.* Here, end systems implement all multicast functionality, including membership management, and packet replication. By eliminating state in routers, and exploiting application-specific intelligence, we argue that End System Multicast can address fundamental concerns with IP Multicast.

We present the design and implementation of protocols for constructing efficient overlays among participating end systems in a self-organizing manner. The scale of nodes involved, and the dynamic and heterogeneous nature of the Internet make the design of these protocols different than traditional distributed algorithms. We present Narada, the first published self-organizing protocol for overlay multicast. We also present Sparta, a protocol deployed in a fully operational broadcasting system based on End System Multicast. The system has been used to broadcast several events including the ACM SIGCOMM and SOSP conferences, and has been used by several thousand users. The thesis adopts an integrated approach to validating architecture, protocol design, and systems building. The protocols address issues such as constructing bandwidth-optimized overlays, and node heterogeneity, that are critical in building operational systems, yet overlooked by the community.

The thesis has influenced the community's thinking on multicast and inspired much follow-on effort. Narada has been extensively used as a benchmark for comparison. Metrics that the thesis introduces, such as *Stress,* and *Relative Delay Penalty,* have become standard benchmarks for evaluating overlay based solutions. Experience gathered from extensive real deployment is a distinguishing highlight of this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The vision driving the research in this thesis is the goal of *ubiquitous* deployment of applications such as audio/video conferencing, distance learning and IP Television over the Internet. "Ubiquitous" refers to enabling anyone with connectivity to the Internet to broadcast his own content, or be able to subscribe to an existing broadcast/conference. Further, this must be enabled in a low-overhead manner and with low cost.

Two key aspects characterize the applications we consider. First, they are *synchronous,* that is, they involve a large number of participants simultaneously tuning into the application. Second, they involve high bandwidth of several hundred or more kilobytes a second. Finally, applications like conferencing are interactive, and may require latencies of less than a few hundred milliseconds.

The conventional wisdom has been that efficiently supporting such applications requires support for multicast functionality at the network level (IP layer). Accordingly, researchers have proposed and studied IP Multicast, an architecture that requires support from routers and network infrastructure. IP Multicast has been widely investigated by the research and industrial community, and these efforts have resulted in numerous publications and Internet standards. However, over a decade after its initial proposal, IP Multicast has remained plagued with concerns related to scalability, network management, and support for higher level functionality like error, flow, and congestion control, and has seen very limited deployment.

This thesis argues that multicast can be *efficiently* supported *without* network level support, and explores this claim in the context of an alternate architecture that we call End System Multicast. End System Multicast is a complete philosophical, and architectural departure from IP Multicast. Here, end systems implement all multicast functionality, including membership management, and packet replication. No support is required from routers, and all complexity is pushed to the edges. By eliminating state in routers, and exploiting application-specific intelligence, we believe that End System Multicast can address fundamental concerns with IP Multicast.

To validate the claims, the thesis has designed and implemented protocols for constructing efficient overlays among participating end systems in a self-organizing manner. The scale of nodes involved, the dynamics of participation (group dynamics and Internet congestion), and the heterogeneity in the Internet (diversity of nodes and diversity of bandwidth and Internet path characteristics) make the design of these protocols very different than traditional distributed algorithms.

The protocols developed by the thesis form a key part of a fully operational broadcasting system based on End System Multicast. The system has been used to broadcast several real events including the ACM SIGCOMM and SOSP conferences, and has been used by over four thousand users. The real deployment and experience obtained help to validate the ESM architecture, have guided the design of ESM prototypes, and form a distinguishing aspect of this thesis.

The rest of this chapter describes the key ideas behind End System Multicast, the benefits and the concerns with the architecture, and an overview of the approach taken by the thesis. We also summarize the contributions made by the thesis and provide a roadmap of the rest of the thesis.

1.1 Background: IP Multicast

IP Multicast is an architecture for group communication proposed by Steve Deering in his seminal paper published in 1989 [15]. Members participating in the application join a multicast group, which is assigned addresses that are part of the Internet Address space. Routers dynamically construct trees for delivering data, which can reach all recipients. Routers are required to maintain state for every individual multicast group, and can replicate packets, and forward them along multiple interfaces.

Multicast was perhaps the first major functionality added to the IP layer since the original design. While more than a decade has passed since IP Multicast was first proposed, several fundamental concerns with the architecture remain unaddressed. As a result, IP Multicast remains practically undeployed to this day.

- *Scalability with number of groups:* IP Multicast requires routers to maintain per group state, which violates the "stateless" architectural principle of the original design. There is no obvious way to aggregate multicast addresses, and routers need to deal with large forwarding state. Further, routers need to maintain per-group control state information, that includes processing periodic per-group control messages, and handling per-group timers. Maintaining such state in a distributed fashion is a challenge.

- *Support of higher layer functionality:* IP Multicast provides a best-effort multi-point delivery service. Support of functionality like reliability and congestion control is left to the end systems. While this separation of routing and transport has worked extremely well in the unicast context, a similar design philosophy has led to several difficulties in the multicast context. Consider the problem of multicast congestion control, where a single sender sends data to multiple receivers. In a heterogeneous Internet environment, each receiver requires transmission of data at a different rate. Sending data at the rate of the slowest receiver results in unnecessary quality degradation of the fastest receiver, and transmitting at the rate of the fastest receiver swamps the slowest machine.

While there have been several innovative ideas for dealing with multicast congestion control, and reliable multicast, these problems are generally acknowledged as hard, and there is no satisfactory solution to these problems yet. For example, an innovative idea for achieving congestion control while multicasting video is layering [64]. The sender encodes video into different layers, where higher layers are refinements over lower layers. Each layer is multicast to a different group. Each receiver subscribes to an appropriate number of layers, depending on its capability. While layered multicast is an interesting idea, several fundamental concerns regarding unfairness, instability, synchronization among receivers, and high latencies involved in adding and dropping multicast groups remain unaddressed [53], [54], [3].

- *Management and deployment issues:* There have been several concerns regarding the deployment of IP Multicast. First, IP Multicast allows any Internet host to send data to a given multicast group, and does not support admission control. This leads to the concern that a malicious host could cause significant disruption to the entire network by flooding data to popular multicast groups. Even a host with limited bandwidth to the Internet can cause significant damage given the multiplicative effect of a multicast-enabled network. Second, while intra-domain multicast is relatively easy to deploy, there are several obstacles to providing support for inter-domain multicast [16]. Third, IP Multicast requires that groups obtain a globally unique address in a consistent and distributed fashion, which is quite hard. Recent work such as Express [29] has attempted to alleviate these concerns, by proposing changes to the IP Multicast model. However, such work is restricted to dealing with large-scale single source applications such as broadcasts, and has met with limited success even in such realms.

1.2 End System Multicast

End System Multicast is based on three key ideas that we summarize below.

- *Push functionality from the routers to the edge:* End System Multicast requires absolutely no change to routers, and all intelligence is implemented at the end systems. The stateless nature of Internet routing is maintained, and the inherent scaling concerns introduced by the IP layer are avoided. Further, deployment is not an issue, as no change is required to network infrastructure.

- *Application-Specific customizability:* IP Multicast provides a generic point-to-multipoint routing abstraction that is independent of any particular application. In contrast, in End System Multicast, we explicitly advocate that the solution be customized to the needs of the individual application. Given the diversity in group communication applications, we believe that a one-size fit-all approach cannot satisfactorily address the requirements of any application.

- *Integrated Multicast Transport and Routing:* IP Multicast attempts to conform to the traditional separation of routing (e.g., IP) and transport (e.g. TCP) that has worked

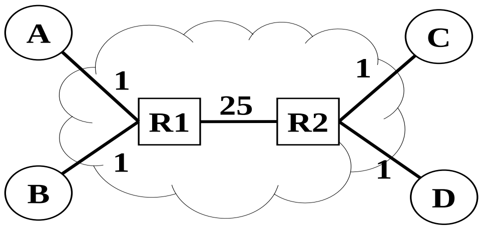

Figure 1.1: Network Configuration

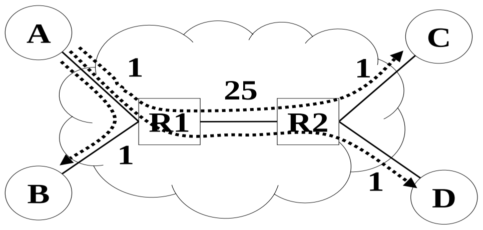

Figure 1.2: Naive Unicast

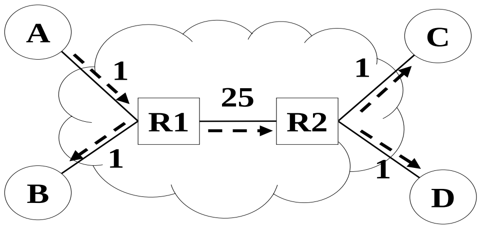

Figure 1.3: IP Multicast

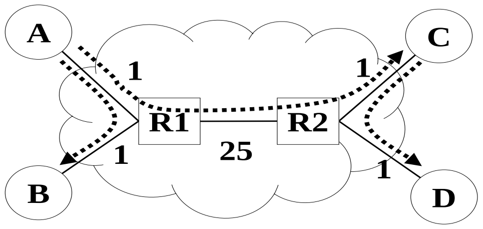

Figure 1.4: End System Multicast

well in the unicast context. However, as we have discussed, such an approach has complicated support of end-to-end reliability, and congestion control in the multicast domain. In contrast, in End System Multicast, we advocate that issues pertaining to routing and transport be treated in an *integrated* fashion. For example, as shown in Figure 1.5, we can tackle receiver heterogeneity by running unicast congestion control algorithms on each overlay link, and by buffering, or application specific packet drop policies (or even transcoding) in splitting nodes of the overlay tree.

Figures 1.1-1.4 illustrate the differences between End System Multicast and IP Multicast. Figure 1.1 depicts an example physical topology. $R1$ and $R2$ are routers, while A, B, C, and D are end systems. Link delays are as indicated. Thus, $R1 - R2$ may be imagined to be a costly transcontinental link, while all other links are cheaper local links. Further, let us assume A wishes to send data to all other members. Figure 1.2 depicts naive unicast transmission. Naive unicast results in significant redundancy on links near the source (for example, link $A - R1$ carries three copies of a transmission by A), and results in duplicate copies on costly links (for example, link $R1 - R2$ has two copies of a transmission by A). Figure 1.3 illustrates how IP Multicast delivers data. Notice that routers can copy packets and forward them along multiple interfaces. Redundant transmission is avoided, and exactly one copy of the packet traverses any given physical link. Figure 1.4 presents an efficient way in which data can be delivered from the sender A to the various receivers, using End System Multicast. The number of redundant copies of data near the source is reduced compared to naive unicast, and just one copy of the packet goes across the costly transcontinental link $R1 - R2$. Yet, this efficiency has been obtained with absolutely no change to routers, and all intelligence is implemented at the end systems.

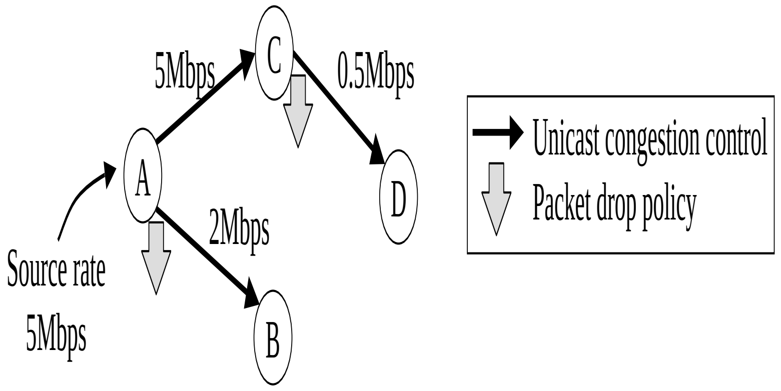

Figure 1.5: Overlay multicast simplifies tackling receiver heterogeneity

1.3 Application End-point Architectures

Given that End System Multicast tries to push functionality to the edges, there are several choices in instantiating such an architecture. On one end of the spectrum is a purely *application end-point* architecture, where functionality is pushed to the end systems actually participating in the multicast group. On the other end of the spectrum is an *infrastructure-centric* architecture, where an organization that provides value added services deploys proxies at strategic locations on the Internet. End systems attach themselves to proxies near them, and receive data using plain unicast.

Both the application end-point, and the infrastructure model are emerging to be important trends in the Internet today. Companies such as Groove Networks [26], Napster [41], and applications like Freenet [13] and Gnutella [25] fall in the application end-point domain, while companies such as Akamai [2], and Fast Forward [30] fall in the infrastructure service model. While both models are interesting in their own right, they offer different constraints and present different sets of challenges.

The focus of this thesis is on an application end-point architecture. This choice was motivated because it was more aligned with our objectives of enabling the *ubiquitous* deployment of broadcasting and conferencing applications. Such an architecture is completely distributed, and leverages the bandwidth resources of end systems actually participating in the group. In contrast, while an infrastructure service can potentially deal with a smaller number of well-defined groups, it is unclear whether it can support the bandwidth requirements associated with deploying tens of thousands of high-bandwidth conferencing and broadcasting applications. Further, an application end-point architecture is instantaneous to deploy, and can enable support of applications with minimal set-up overhead and low cost.

While application end-point architectures have the promise to enable ubiquitous deployment, infrastructure-centric architecture can potentially provide more robust data delivery. Infrastructure-centric architectures involve better provisioned proxy hosts that can be located at strategic locations on the Internet. Further, the probability of failure of such hosts is low. In contrast, application end-point architectures potentially involve a wider range of end systems that may not provide as good performance, and must deal with the failures of end systems. Finally, infrastructure-based architectures may scale better to very large group sizes given the better robustness properties they provide.

Infrastructure-centric and application end-point architectures represent two ends of a rich spectrum. One can conceive of several hybrid architectures that both leverage resources at end system participants, as well as infrastructure nodes. While we do not explore such architectures in this thesis, we note that our deployment efforts described in Chapter 4 employ additional hosts that we call waypoints - the objective is to construct overlays using actual end systems to the extent possible, but to gracefully leverage waypoints to enhance performance when available.

In the rest of this dissertation, by End System Multicast, we implicitly refer to an application end-point architecture.

1.4 Concerns with ESM

While End System Multicast has the potential to address many of the issues with IP Multicast, a key concern with the architecture is performance. The performance concerns stem from several reasons:

- *Fundamental inefficiencies of overlays:* An overlay approach cannot be as efficient as an approach involving native support from IP Multicast. ESM cannot completely eliminate the transmission of redundant packets on physical links, and potentially involves higher latencies than IP Multicast. For example, in Figure 1.4, the latency from A to D has been increased using ESM as compared to using IP Multicast, while the link $A-B$ carries redundant transmissions.

- *Limited knowledge of network* Efficient overlays must be constructed in a diverse and heterogeneous environment. Further, this must be done by members that have a limited view of the network, and potentially using distributed approaches.

- *Dynamic:* Efficient overlays must be constructed in a dynamic environment where members may join and leave dynamically, members may fail, and issues such as congestion in Internet links must be dealt with.

In addition to the performance concerns, there are unknowns regarding how willing users are to cooperate in the overlay construction and donate their bandwidth resources. Finally, there are concerns regarding malicious clients that may disrupt data delivery. In this thesis however, we will primarily focus on the performance concerns with End System Multicast.

1.5 Thesis Approach and Discussion

This thesis research is primarily motivated by the question: is it viable to provide good performance to applications using an End System Multicast architecture?

To gain insight into the question, the thesis has designed and implemented protocols for constructing efficient overlays among participating end systems in a self-organizing manner. The scale of nodes involved, the dynamics of participation (group dynamics and Internet congestion), and the heterogeneity in the Internet (diversity of nodes and diversity of bandwidth and Internet path characteristics) make the design of these protocols very different than traditional distributed algorithms.

The protocols designs have been motivated by real applications, and have been deployed in a fully operational broadcasting system based on End System Multicast. The wide-spread operational use of the broadcasting system has helped to validate the End System Multicast architecture.

The thesis has adopted an integrated approach to validating ESM architecture, design of protocols and system building. The protocols have been designed with the view that they represent prototype solutions that help to investigate the feasibility of the ESM architecture. Building a full-fledged operational system has influenced an empirical approach to protocol design. The thesis identifies important real-world issues that must be addressed by protocol designs, considers techniques to tackle the issues, and empirically demonstrates the effectiveness of the techniques.

The thesis has attempted to demonstrate the performance potential of End System Multicast in several phases. Each phase has involved an increase in the realism of the effort, and has built on and extended earlier designs. The initial prototypes developed were evaluated using simulations. The thesis then moved to experiments with the prototypes in Internet test-bed environments, during which key refinements were added relating to adaptation to Internet dynamics and light-weight probing heuristics. The designs so developed were adapted into real deployment. Experience with a deployed system and users both further helped establish the credibility of ESM, and has provided access to real workloads and insights that are guiding future iterations of the design process.

1.6 Contributions

We summarize the key contributions of the thesis below:

Architecture: The primary contribution of this thesis is demonstrating that *"it is feasible to efficiently enable a wide range of group communication applications using an End System Multicast architecture."* This thesis is demonstrated in the context of two important bandwidth-demanding applications - audio/video conferencing and broadcasting. The thesis is validated through extensive simulation and Internet test-bed experiments, as well as live deployment of a real broadcasting system based on ESM which has been used to support several thousand users. The thesis work has played a key role in the emergence of overlay multicast as a field of its own right [24, 11, 34, 10, 36, 49, 66, 37, 57, 43, 73, 14, 8], and has influenced the community's thinking on the right architecture for multicast.

Protocol Design The thesis has designed and implemented two self-organizing protocols for End System Multicast. The protocols construct efficient overlays among participating end systems in a distributed and self-improving fashion, taking user dynamics into account. The specific protocol designs include:

- **Narada,** a protocol motivated by multi-source conferencing applications. Narada is the first published self-organizing protocol for overlay multicast, and has been used extensively as a baseline of comparison by the research community [57].
- **Sparta,** a protocol for single-source broadcasting applications that forms part of the operationally deployed broadcasting system. This is perhaps the only self-organizing protocol in the community that has seen real user experience.

Investigation of real-world issues that impact protocol design: The objective of building a fully operational system has influenced a unique systems approach to the protocol designs that distinguishes the thesis research from other concurrent efforts in the community. Some specific contributions are:

- The thesis perhaps provides the most extensive experience with handling bandwidth as a metric in overlay construction, while most protocols in literature typically consider delay-based metrics. The contributions include a study that systematically demonstrates the importance and viability of optimizing overlays for dual metrics - bandwidth and latency - in contructing overlays targeted for conferencing applications. The study has also evaluated light-weight probing heuristics while constructing overlays optimized for bandwidth.
- The thesis has conducted a study of the impact of heterogeneity in outgoing bandwidth constraints of nodes on protocols of two different classes of design for End System Multicast. The issue has been motivated based on our deployment experience, and our study is perhaps the first systematic investigation of the issue to the best of our knowledge.

System Artifacts: The protocol designs in this thesis have evolved through implementation, and form a key component of a full-fledged broadcasting system based on End System Multicast. The system is satisfying the needs of real content publishers and viewers. It has been used to broadcast tens of events, and has been used by several thousand users. The extensive and sustained usage of the system are proving it to be an important contribution in its own right.

Metrics and methodology: This thesis makes key contributions to metrics and methodology for evaluating ESM prototypes. For example, metrics such as *Stress* and *Relative Delay Penalty* introduced by this thesis have been extensively used in the research community to evaluate overlay protocols.

1.7 Relation to other theses

Much of this thesis research has been conducted in conjunction with a companion thesis [1]. Both theses are motivated by the primary objective of demonstrating the viability of the

End System Multicast architecture, have made significant contributions to the building of the ESM broadcasting system, and share much of the evaluation metrics and methodology.

The theses differ in the issues emphasized towards the common goal of validating the ESM architecture. This thesis has emphasized issues related to self-organizing protocols for ESM. It has led the design and implementation of the Narada and Sparta protocols, and has conducted a study of the impact of heterogeneous out-going bandwidth constraints of nodes on protocol design. In contrast, [1] has emphasized issues in the design of a system that are orthogonal to protocol design. The contributions of [1] include the design of APIs between the application and ESM software, handling variations in receiver capabilities through application-level adaptation, and incentive mechanisms for users.

1.8 Thesis Organization

The rest of the thesis is organized as follows.

Chapter 2 presents the design of the Narada protocol. Narada was targeted at small-scale multi-source applications such as conferencing. In the context of the Narada protocol, the chapter studies key performance concerns with End System Multicast using a simulation study.

Chapter 3 investigates the viability of End System Multicast to support bandwidth-demanding applications in real Internet settings. The focus is on conferencing applications that simultaneously need high bandwidth and low latencies. The chapter presents techniques that enable Narada to build overlays that are simultaneously optimized for both metrics. A detailed evaluation on Internet-testbeds demonstrates the importance of adapting to both metrics and shows that the heuristics work well.

Chapter 4 describes experience with a fully operational broadcasting system based on End System Multicast. This thesis is responsible for the design and implementation of Sparta - the protocol used in the system. While Narada was targeted at smaller-scale multi-source conferencing applications, the system was targeted at larger-scale single-source broadcasting applications. This has led to a protocol that has a different architecture than Narada, yet which builds on several insights obtained with the initial Narada prototype. The chapter presents the details of Sparta, the deployment of the system, a detailed characterization of the performance, and lessons learnt through the experience. Overall, the results show that ESM is easy to deploy and can provide good application performance.

Chapter 5 presents a systematic study of a key issue that was highlighted by our deployment experience - the need to explicitly consider the heterogeneity in node characteristics in the design of ESM protocols. The implications of this issue is studied for two different classes of protocol designs for End System Multicast.

Chapter 6 presents related work, and Chapter 7 presents conclusions and future research directions.

Chapter 2

The Narada Protocol:Design and Evaluation

In this chapter, we present Narada, a protocol that constructs efficient data delivery trees among participating members in a self-organizing manner. The design of Narada has been influenced by smaller-scale multi-source applications such as audio/video conferencing. This chapter presents the generic framework of Narada, but key issues in customizing Narada to bandwidth demanding applications are discussed in Chapter 3.

Narada constructs an overlay structure among participating end systems in a *self-organizing* and *fully distributed* manner. Narada is robust to the failure of end systems and to dynamic changes in group membership. End systems begin with no knowledge of the underlying physical topology, and they determine latencies to other end systems by probing them in a controlled fashion. Narada continually refines the overlay structure as more probe information is available.

We study two key performance concerns relating to End System Multicast in the context of the Narada protocol: (i) how effective is an overlay architecture in minimizing the transmission of redundant packets on physical links?; and (ii) what delay penalties do receivers observe using an overlay-based approach? Our detailed simulation results with Narada are encouraging. In a group of 128 members, the delay between at least 90% of pairs of members increases by a factor of at most 4 compared to the unicast delay between them. Further, no physical link carries more than 9 identical copies of a given packet.

The rest of the chapter is organized as follows. Section 2.1 presents the design goals with Narada, the key design decisions taken, and the alternatives. Section 2.2 presents a detailed description of the Narada protocol. Sections 2.3 and 2.4 present our simulation methodology and results, and we conclude in Section 2.5.

2.1 Narada Design

In designing Narada, we have the following objectives in mind:

• *Self-organizing:* The construction of the end system overlay must take place in a fully distributed fashion and must be robust to dynamic changes in group membership.
• *Overlay efficiency:* The constructed overlay must have a low redundancy on underlying physical links, and delays from the source to each receiver must be low. Further, the out-

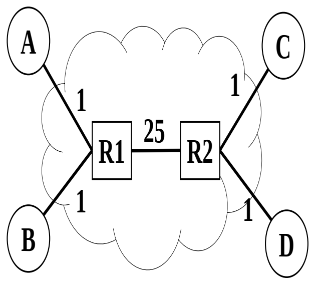

Figure 2.1: Network Topology

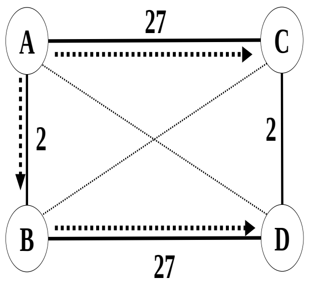

Figure 2.2: Illustration of the mesh-first approach used by Narada

degree of each member in the overlay (or the number of children a member can sustain) must reflect the bandwidth of its connection to the Internet.

• *Self-improving in an incremental fashion:* The overlay construction must include mechanisms by which end systems gather network information in a scalable fashion. The protocol must allow for the overlay to incrementally evolve into a better structure as more information becomes available.

Design Space: Narada constructs an overlay tree rooted at each source for delivering data from that source. There are two possible design alternatives. The first alternative is to have no formal data delivery structure, and to have every data packet distributed using epidemic or gossip-like algorithms. Such an approach can be inefficient with a host potentially getting multiple copies of the same data packet. The second alternative is to construct more resilient structures than trees. The naive approach to this can have a high overhead - recent follow-up work on Narada has investigated constructing redundant structures in conjunction with specialized coding algorithms as a way to controlling the overhead [43, 8].

Narada is targeted at multi-source applications (e.g. audio/video conferencing). While the goal is to construct trees rooted at each source, it does not do so directly but in a two-step process. In the first step, it constructs a richer connected graph that we term *mesh*. The mesh could in general be an arbitrary connected subgraph of the complete graph of the participating nodes, but Narada constructs a mesh with has desirable performance properties as discussed later. In the second step, Narada constructs (reverse) shortest path spanning trees of the mesh, each tree rooted at the corresponding source using well known routing algorithms. Figure 2.2 presents an example mesh that Narada constructs for the physical topology shown in Figure 2.1, along with the shortest path spanning tree rooted at A.

In our approach, given that data delivery trees are constructed from among links of the mesh, it is important to construct a good mesh so that good quality trees may be produced. In particular, we attempt to ensure the following properties: (i) the shortest path delay between any pair of members along the mesh is at most K times the unicast delay between them, where K is a small constant and (ii) each member has a limited number of neighbors in the mesh which does not exceed a given (per-member) bound chosen to reflect the bandwidth of the member's connection to the Internet.[1] Limiting the number of neighbors regulates the fanout of members in the spanning trees. Second, it controls the overhead of running routing algorithms on the mesh. The extreme case where the mesh is chosen to be the complete graph of all nodes and all pairs of links connecting them incurs all the overhead of routing with none of its benefits as the resulting shortest path spanning trees degenerates to naive unicast transmission.

There are two natural design alternatives to the mesh-based approach that we adopt:

• *Shared Trees:* A first alternative is to construct a single shared tree rather than a tree for each source. While shared trees are potentially not as efficient as per-source trees, a greater difficulty is maintaining connectivity of the tree structure. Typically shared tree approaches involve needing to elect a root of the shared tree - handling failure of the root is not easy and it is difficult to analyze the robustness properties of the resulting designs.
• *Multiple independently constructed trees:* A second alternative is to construct a tree for each source, with each tree constructed directly in that members select parents from among the members they know. Such an approach requires seperate algorithms for group management (having members know about other members), and also require mechanisms by which a new member joining the group and becoming a source can announce itself, so that existing members can join the tree rooted at this new member. In contrast, in the mesh-based approach of Narada, group management functions are abstracted out and handled at the mesh, and we may leverage standard routing algorithms for construction of data delivery trees.

2.2 Design Details

In this section, we present a detailed description of Narada. We explain the distributed algorithms that Narada uses to construct and maintain the mesh in Section 2.2.1. We present heuristics that Narada uses to improve mesh quality in Section 2.2.2. Narada runs a variant of standard distance vector algorithms on top of the mesh and uses well known algorithms to construct per-source (reverse) shortest path spanning trees for data delivery. We discuss this in Section 2.2.3.

2.2.1 Group Management

We have seen that Narada constructs a mesh among end systems participating in the multicast group. In this section, we present mechanisms Narada uses to keep the mesh connected,

[1]An ideal mesh is a "Degree-Bounded K-spanner" [35] of the complete graph of all nodes in the group and all links connecting them. The problem of constructing Degree-Bounded K-spanners of a graph has been widely studied in centralized settings that assume complete information and is NP-complete even in such scenarios [35].

```
Let $i$ receive refresh message from neighbor $j$ at $i's$ local
time $t$. Let $< k, s_{kj} >$ be an entry in $j's$ refresh message.
• if $i$ does not have an entry for $k$, then $i$ inserts the
          entry $< k, s_{kj}, t >$ into its table
• else if $i$'s entry for $k$ is $< k, s_{ki}, t_{ki} >$, then
    • if $s_{ki} \geq s_{kj}$ $i$ ignores the entry pertaining to $k$
    • else $i$ updates its entry for $k$ to $< k, s_{kj}, t >$
```

Figure 2.3: Actions taken by a member i on receiving a refresh message from member j.

to incorporate new members into the mesh and to repair possible partitions that may be caused by members leaving the group or by member failure.

As we do not wish to rely on a single non-failing entity to keep track of group membership, the burden of group maintenance is shared jointly by all members. To achieve a high degree of robustness, our approach is to have every member maintain a list of all other members in the group. Since Narada is targeted towards small sized groups, maintaining the complete group membership list is not a major overhead. Every member's list needs to be updated when a new member joins or an existing member leaves. The challenge is to disseminate changes in group membership efficiently, especially in the absence of a multicast service provided by the lower layer. We tackle this by exploiting the mesh to propagate such information. However, this strategy is complicated by the fact that the mesh might itself become partitioned when a member leaves. To handle this, we require that each member periodically generate a refresh message with monotonically increasing sequence number, which is disseminated along the mesh. Each member i keeps track of the following information for every other member k in the group: (i) member address k; (ii) last sequence number s_{ki} that i knows k has issued; and (iii) *local time* at i when i first received information that k issued s_{ki}. If member i has not received an update from member k for T_m time, then, i assumes that k is either dead or potentially partitioned from i. It then initiates a set of actions to determine the existence of a partition and repair it if present as discussed in Section 2.2.1.3.

Propagation of refresh messages from every member along the mesh could potentially be quite expensive. Instead, we require that each member periodically exchange its knowledge of group membership with its neighbors in the mesh. A message from member i to a neighbor j contains a list of entries, one entry for each member k that i knows is part of the group. Each entry has the following fields: (i) member address k; and (ii) last sequence number s_{ki} that i knows k has issued. On receiving a message from a neighbor j, member i updates its table according to the pseudo code presented in Figure 2.3.

Finally, given that a distance vector routing algorithm is run on top of the mesh (Section 2.2.3), routing update messages exchanged between neighbors can include member sequence number information with minimum extra overhead.

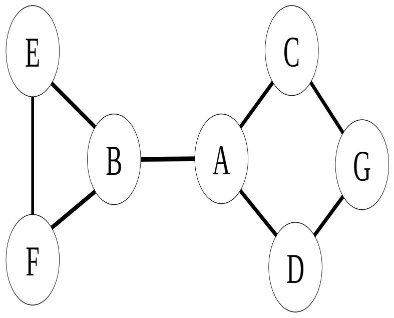

Figure 2.4: A sample virtual topology

2.2.1.1 Member Join

When a member wishes to join a group, Narada assumes that the member is able to get a list of group members by an out-of-band bootstrap mechanism. The list needs neither be complete nor accurate, but must contain at least one currently active group member. In this paper, we do not address the issue of the bootstrap mechanism. We believe that such a mechanism is application specific and our protocol is able to accommodate different ways of obtaining the bootstrap information.

The joining member randomly selects a few group members from the list available to it and sends them messages requesting to be added as a neighbor. It repeats the process until it gets a response from some member, when it has successfully joined the group. Having joined, the member then starts exchanging refresh messages with its neighbors. The mechanisms described earlier will ensure that the newly joined member and the rest of the group learn about each other quickly.

2.2.1.2 Member Leave and Failure

When a member leaves a group, it notifies its neighbors, and this information is propagated to the rest of the group members along the mesh. In Section 2.2.3, we will describe our enhancement to distance vector routing that requires a leaving member to continue forwarding packets for some time to minimize transient packet loss.

We also need to consider the difficult case of abrupt failure. In such a case, failure should be detected locally and propagated to the rest of the group. In this paper, we assume a failstop failure model [61], which means that once a member dies, it remains in that state, and the fact that the member is dead is detectable by other members. We explain the actions taken on member death with respect to Figure 2.4. This example depicts the mesh between group members at a given point in time. Assume that member C dies. Its neighbors in the mesh, A, G stop receiving refresh messages from C. Each of them independently send redundant probe messages to C, such that the probability every probe message (or its reply) is lost is very small. If C does not respond to any probe message, then, A and G assume C to be dead and propagate this information throughout the mesh.

Every member needs to retain entries in its group membership table for dead members. Otherwise, it is impossible to distinguish between a refresh announcing a new member and

```
Let Q be a queue of members for which $i$ has stopped
receiving sequence number updates for at least $T_m$
time. Let $T$ be maximum time an entry may remain in Q.
while(1) begin
     Update Q;
     while( $!Empty(Q)$ and
            $Head(Q)$ is present in Q for $\geq$ $T$ time)
     begin
        $j$= $Dequeue(Q)$;
        Initiate probe cycle to determine if $j$ is dead
        or to add a link to it.
     end
     if( $!Empty(Q)$) begin
        $prob$ = $Length(Q)$/ $GroupSize$;
        With probability $prob$ begin
           $j$= $Dequeue(Q)$;
           Initiate probe cycle to determine if $j$ is dead
           or to add a link to it.
        end
      sleep(P). // Sleep for time P seconds
 end
```

Figure 2.5: Scheduling algorithm used by member i to repair mesh partition

a refresh announcing stale information regarding a dead member. However, dead member information can be flushed after sufficient amount of time.

2.2.1.3 Repairing Mesh Partitions

It is possible that member failure can cause the mesh to become partitioned. For example, in Figure 2.4, if member A dies, the mesh becomes partitioned. In such a case, members must first detect the existence of a partition, and then repair it by adding at least one virtual link to reconnect the mesh. Members on each side of the partition stop receiving sequence number updates from members on the other side . This condition is detected by a timeout of duration T_m.

Each member maintains a queue of members that it has stopped receiving sequence number updates from for at least T_m time. It runs a scheduling algorithm that periodically and probabilistically deletes a member from the head of the queue. The deleted member is probed and it is either determined to be dead, or a link is added to it. The scheduling algorithm is adjusted so that no entry remains in the queue for more than a bounded period of time. Further, the probability value is chosen so that in spite of several members simultaneously attempting to repair partition only a small number of new links are added. The algorithm is summarized in Figure 2.5.

```
EvaluateUtility ($j$) begin
utility = 0
for each member $m$ ($m$ not $i$) begin
    $CL$ = current latency between $i$ and $m$ along mesh
    $NL$ = new latency between $i$ and $m$ along mesh
                 if edge $i$-$j$ were added
       if ($NL < CL$) then begin
           $utility \mathrel{+}= \frac{CL-NL}{CL}$
       end
 end
 return $utility$
```

Figure 2.6: Algorithm i uses in determining utility of adding link to j

2.2.2 Improving mesh quality

The constructed mesh can have a poor quality because: (i) initial neighbor selection by a member joining the group is random given limited availability of topology information at bootstrap; (ii) partition repair might aggressively add edges that are essential for the moment but not useful in the long run; (iii) group membership may change due to dynamic join and leave; and (iv) underlying network conditions, routing and load may vary. Narada allows for incremental improvement of mesh quality. Members probe each other at random and new links may be added depending on the perceived gain in *utility* in doing so. Further, members continuously monitor the utility of existing links, and drop links perceived as not useful. This dynamic adding and dropping of links in the mesh distinguishes Narada from other topology maintenance protocols.

The issue then is the design of a utility function that reflects mesh quality. A good quality mesh must ensure that the shortest path delay between any pair of members along the mesh is comparable to the unicast delay between them. A member i computes the utility gain if a link is added to member j based on (i) the number of members to which j improves the routing delay of i; and (ii) how significant this improvement in delay is. Figure 2.6 presents pseudo code that i uses to compute the gain in utility if a link to member j is added. The utility can take a maximum value of n, where n is the number of group members i is aware of. Each member m can contribute a maximum of 1 to the utility, the actual contribution being $i's$ relative decrease in delay to m if the edge to j were added. We now present details of how Narada adds and removes links from the mesh.

Addition of links: Narada requires every member to constantly probe other members. Currently, the algorithm that we use is to conduct a probe periodically, and probe some random member each time. This algorithm could be made smarter by varying the interval between probes depending on how satisfied a member is with the performance of the mesh, as well as choosing whom to probe based on results of previous probes.

When a member i probes a member j, j returns to i a copy of its routing table. i uses this information to compute the gain in utility if a link to j is added as described in Figure 2.6. i decides to add a link to j if the utility gain exceeds a given threshold. The

```
ShouldAddNeighbor (j) begin
    utilityGain = EvaluateUtility(j)
    thresh = $\frac{GroupSize}{(MaxFanout(i) * m)}$
     /* m is a damping constant $\geq$ 1.0 */
    if (CurrentFanout(j) < MinFanout(j)) and
       (CurrentFanout(i) < MinFanout(i)) then begin
             thresh /= 2;  /* lower thresh if current
                                   fanout of i, j low */
       end
    if (utilityGain > thresh) return yes;
    CL = current latency between i and j along mesh
    PL = physical latency between i and j
    /* Nodes with latency < L are considered close */
    /* K is highest tolerable penalty for close nodes */
    if(PL < L and $\frac{CL}{PL}$ > K and
       CurrentFanout(i) < MaxFanout(i)  and
       CurrentFanout(j) < MaxFanout(j))
           return yes;
     else
           return no;
end
```

Figure 2.7: Algorithm i uses in determining if a link must be added to j

threshold value is a function of $i's$ estimation of group size, and the current and maximum fanout values of i and j respectively. Finally, i may also add a link to j if the physical delay between them is very low and the current overlay delay between them very high.

Dropping of links: Ideally, the loss in utility if a link were to be dropped must exactly equal the gain in utility if the same link were immediately re-added. However, this requires estimating the relative increase in delay to a member if a link were dropped and it is difficult to obtain such information. Instead, we overestimate the actual utility of a link by its *cost*. The cost of a link between i and j in $i's$ perception is the number of group members for which i uses j as next hop. Periodically, a member computes the *consensus cost* of its link to every neighbor using the algorithm shown in Figure 2.8. It then picks the neighbor with lowest consensus cost and drops it if the consensus cost falls below a certain threshold. The threshold is again computed as a function of the member's estimation of group size and its current and maximum fanout. The consensus cost of a link represents the maximum of the cost of the link in each neighbor's perception. Yet, it might be computed locally as the mesh runs a distance vector algorithm with path information.

Our heuristics for link-dropping have the following desirable properties:

• *Stability:* A link that Narada drops is unlikely to be added again immediately. This is ensured by several factors: (i) the threshold for dropping a link is less than or equal to the threshold for adding a link; (ii) the utility of an existing link is overestimated by the cost metric; (iii) dropping of links is done considering the perception that both members have

```
EvaluateConsensusCost(j) begin
 $Cost_{ij}$ = number of members for which $i$ uses $j$ as
             next hop for forwarding packets.
 $Cost_{ji}$ = number of members for which $j$ uses $i$ as
             next hop for forwarding packets.
 return  max($Cost_{ij}$, $Cost_{ji}$)
end
```

Figure 2.8: Algorithm i uses to determine consensus cost to a neighbor j

regarding link cost; (iv) a link with small delay is not dropped.

•*Partition avoidance:* We present an informal argument as to why our link dropping algorithm does not cause a partition assuming steady state conditions and assuming multiple links are not dropped concurrently. Assume that member i drops neighbor j. This could result in at most two partitions. Assume the size of $i's$ partition is S_i and the size of $j's$ partition is S_j. Further, assume both i and j know all members currently in the group. Then, the sum of S_i and S_j is the size of the group. Thus $Cost_{ij}$ must be at least S_j and $Cost_{ji}$ must be at least S_i, and at least one of these must exceed half the group size. As long as the drop threshold is lower than half the group size, the edge will not be dropped.

2.2.3 Data Delivery

We have described how Narada constructs a mesh among participating group members, how it keeps it connected, and how it keeps refining the mesh. In this section we explain how Narada builds data delivery tree.

Narada runs a distance vector protocol on top of the mesh. In order to avoid the well-known count-to-infinity problems, it employs a strategy similar to BGP [51]. Each member not only maintains the routing cost to every other member, but also maintains the path that leads to such a cost. Further, routing updates between neighbors contains both the cost to the destination and the path that leads to such a cost. The per-source trees used for data delivery are constructed from the reverse shortest path between each recipient and the source, in identical fashion to DVMRP [15]. A member M that receives a packet from source S through a neighbor N forwards the packet only if N is the next hop on the shortest path from M to S. Further, M forwards the packet to all its neighbors who use M as the next hop to reach S.

The routing metric used in the distance vector protocol is the latency between neighbors. Each endpoint of a link independently estimates the latency of the link and could have different estimates. Using the latency as a metric enables routing to adapt to dynamics in the underlying network. However, it also increases the probability of routing instability and oscillations. In our work, we assume that members use an exponential smoothing algorithm to measure latency. Further, the latency estimate is updated only at periodic intervals. The period length can be varied to tradeoff routing stability with reactivity to changing conditions.

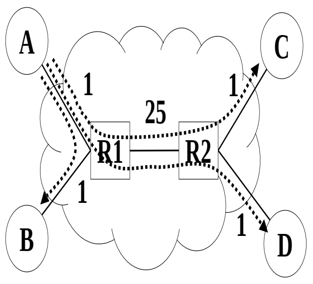

Figure 2.9: Naive Unicast

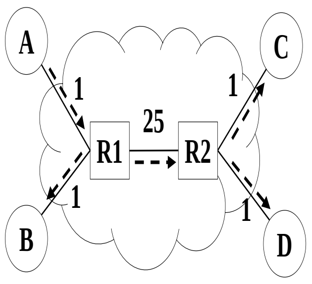

Figure 2.10: IP Multicast

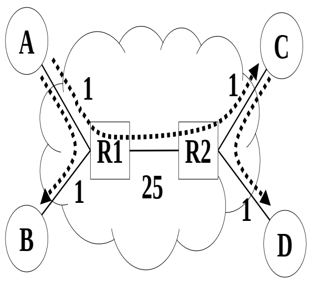

Figure 2.11: ESM

A consequence of running a routing algorithm for data delivery is that there could be packet loss during transient conditions when member routing tables have not yet converged. In particular, there could be packet loss when a member leaves the group or when a link is dropped for performance reasons. To avoid this, data continues to be forwarded along old routes for enough time until routing tables converge. To achieve this, we introduce a new routing cost called *Transient Forward (TF)*. TF is guaranteed to be larger than the cost of a path with a valid route, but smaller than infinite cost. A member M that leaves advertises a cost of *TF* for all members for which it had a valid route. Normal distance vector operations leads to members choosing alternate valid routes not involving M (as *TF* is guaranteed to be larger than the cost of any valid route). The leaving member continues to forward packets until it is no longer used by any neighbor as a next hop to reach any member, or until a certain time period expires.

2.3 Simulation Experiments

In this section, we present our methodology for evaluating the properties of the overlay structure that Narada produces, and the overheads associated with the Narada protocol.

2.3.1 Performance Indices

An overlay structure fundamentally cannot perform as efficiently as IP Multicast. We are interested in evaluating the quality of the structure produced by Narada and in comparing it with two alternate methods of data dissemination: IP Multicast using DVMRP[15] and naive unicast. Figures 2.9, 2.10 and 2.11 show dissemination structures using unicast, IP Multicast, and End System Multicast.

To facilitate our comparison, we consider the following metrics:

• *Relative Delay Penalty (RDP)* , which is a measure of the increase in delay that applications perceive while using ESM. For example, in Figure 2.11, the delay from A to D with ESM is 29, the delay of the unicast path from A to D is 27, and the RDP of <A,D> is $\frac{29}{27}$.

• *Worst Case Stress* , defined as $\max_{i=1}^{L} s_i$, where L is the number of physical links used in transmission and s_i is the stress of link i. The stress of a physical link is the number of identical copies of a packet carried by the link. In Figure 2.11, link $R1 - R2$ has a stress of 1, link $A - R1$ has a stress of 2, and the worst case stress is 2.
• *Normalized Resource Usage (NRU)* , defined as the ratio of the resource usage with Narada relative to resource usage of DVMRP. We define resource usage as $\sum_{i=1}^{L} d_i * s_i$, where, L is the number of links active in data transmission, d_i is the delay of link i and s_i is the stress of link i. The resource usage is a metric of the network resources consumed in the process of data delivery to all receivers, and the NRU is a measure of the additional network resources consumed by ESM compared to IP Multicast. Implicit here is the assumption that links with higher delay tend to be associated with higher cost. The resource usage metric does not affect performance perceived by the application, however it measures how effectively an overlay topology makes use of network resources, for instance by clustering receivers that are located near each other. The resource usage might be computed to be 30 in the case of transmission by DVMRP, 57 for naive unicast and 32 for the smarter tree, shown in Figure 2.10, Figure 2.9 and Figure 2.11 respectively, and the NRU for ESM is $\frac{32}{30}$.

DVMRP has an RDP of 1 (assuming symmetric routing), a worst case stress of 1 and by definition an NRU of 1. Naive unicast has an RDP of 1 (by definition) and a worst case stress of r, when r is the number of receivers.

Finally, we also evaluate the time it takes for the overlay to stabilize and the protocol overhead that Narada introduces. In this chapter, we do not consider performance metrics related to behavior under transient conditions, such as packet loss.

2.3.2 Factors that affect Narada's Performance

We have investigated the effects of the following factors on Narada's performance: (i) topology model; (ii) topology size; (iii) group size and (iv) fanout range.

We used three different models to generate backbone topologies for our simulation. For each model of the backbone, we modeled members as being attached directly to the backbone topology. Each member was attached to a random router, and was assigned a random delay of $1 - 4ms$.

• *Waxman:* The model considers a set of n vertices on a square in the plane and places an edge between two points with a probability of $\alpha e^{\frac{-d}{\beta * L}}$, where, d is the distance between vertices , L is the length of the longest possible edge and α and β are parameters. We use the Georgia Tech. [75] random graph generators to generate topologies of this model.
• *Mapnet:* Backbone connectivity and delay are modeled after actual ISP backbones that could span multiple continents. Connectivity information is obtained from the CAIDA Mapnet project database [23]. Link delays are assigned based on geographical distance between nodes.
• *Automous System map (ASMap):* Backbone connectivity information is modeled after inter-domain Internet connectivity. This information is collected by a route server from

BGP routing tables of multiple geographically distributed routers with BGP connections to the server [22]. This data has been analyzed in [20] and has been shown to satisfy certain power laws. Random link delays of $8-12$ ms was assigned to each physical link.

In our simulations, we used backbone topology sizes consisting of up to 1070 members and multicast groups of up to 256 members. The fanout range of a member is the minimum and maximum number of neighbors each member strives to maintain in the mesh. An increase of the fanout range could decrease mesh diameter and result in lower delay penalties. However, it could potentially result in higher stress on links near members.

In addition, we identify network routing policy and group distribution as factors that could impact Narada's performance but do not investigate these in this chapter. Routing policy could be significant because in the event that routing is not based on shortest path, some pairs of members could have an RDP of less than 1 with Narada. Group distribution is important as presence of clusters in groups could improve Narada's performance compared to unicast. This is because Narada could minimize the number of copies of a packet that enter a cluster via costlier inter-cluster links and distribute them along cheaper intra-cluster links.

2.3.3 Simulation Setup

We use a locally written, packet-level, event-based simulator to evaluate our protocol. The simulator assumes shortest delay routing between any two members. The simulator models the propagation delay of physical links but does not model queueing delay and packet losses. This was done to make our simulations more scalable. We consider dynamic network conditions by presenting experiments on the Internet in later chapters of this thesis.

All experiments we report here are conducted in the following manner. A fixed number of members join the group in the first 100 seconds of the simulation in random sequence. A member that joins is assumed to contain a list of all members that joined the group previously. After 100 seconds, there is no further change in group membership. One sender is chosen at random to multicast data at a constant rate. We allow the simulation to run for 40 minutes. In all experiments, neighbors exchanges routing messages every 30 seconds. Each member probes one random group member every 10 seconds to evaluate performance.

2.4 Results

We begin by presenting results from a typical experiment that characterizes key aspects of Narada's performance in Section 2.4.1. In Section 2.4.2, we present results that investigate the influence of the factors on Narada's performance. We do not adopt a full factorial design that investigates every possible combination of all factors. Instead we study the influence of each individual factor on Narada's performance one at a time, keeping other factors fixed. We present protocol overhead incurred with Narada in Section 2.4.3. Finally, we summarize and interpret our results in Section 2.4.4.

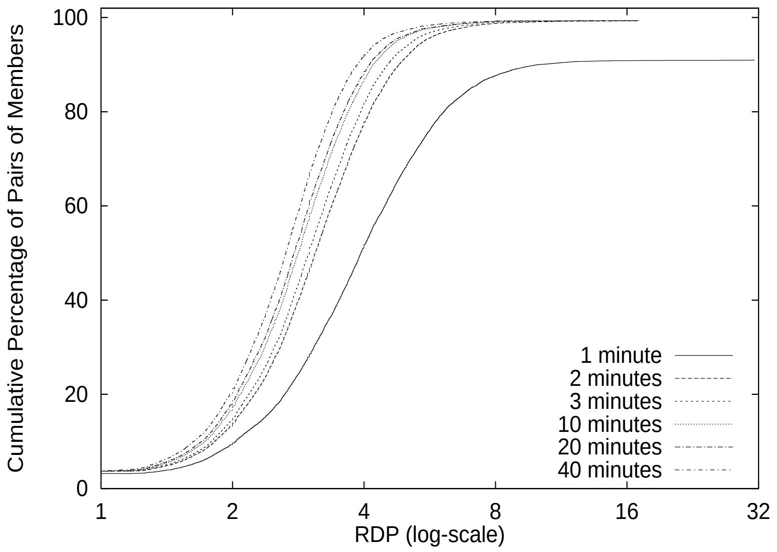

Figure 2.12: Cumulative distribution of RDP shown at various snapshots of the simulation. The minutes denote the time after the last join.

2.4.1 Simulation Results with a Typical Run

This section presents results from a single typical experiment. The results are typical in the sense they capture some of the key invariants in the performance of Narada across all runs. In the experiment, we used a topology generated by the Waxman model consisting of 1024 nodes and 3145 links. We used a group size of 128 members, and each member had a fanout range of <3-6>.

Delay Penalty and Stabilization Time: Figure 2.12 plots the cumulative distribution of RDP at different time instances during the simulation. The horizontal axis represents a given value of RDP and the vertical axis represents the percentage of pairs of group members for which the RDP was less than this value. Each curve corresponds to the cumulative distribution at a particular time instance. It might happen that at a given time, some members have not yet learned of the existence of some other members or do not have routes to others. Thus, 1 minute after the last join, approximately 10% of pairs do not have routes to each other, indicated by the lower curve. All pairs have routes to each other 2 minutes after the last join. As time increases, the curve moves to the left, indicating the RDP is reduced as the quality of the overlay improves.

When the system stabilizes, about 90% of pairs of members have RDP less than 4. However, there exist a few pairs of members with high RDP. This tail can be explained from Figure 2.13. Each dot in this figure indicates the existence of a pair of members with a given RDP and physical delay. We observe that all pairs of members with high RDP have very small physical delays. Such members are so close to each other in the physical network that even a slightly sub-optimal configuration leads to a high delay penalty. However, the

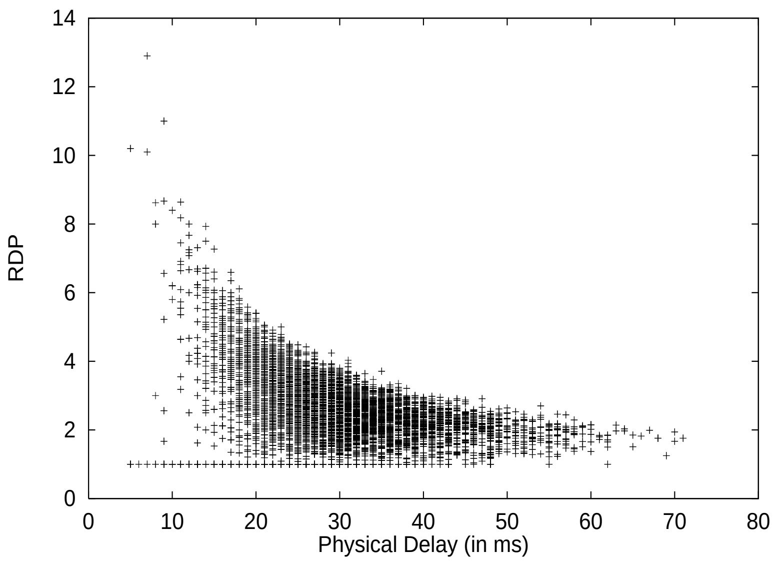

Figure 2.13: RDP vs. physical delay. Each point denotes the existence of a pair of members with a given physical delay and RDP

delay between them along the overlay is not too high. This can be seen from Figure 2.14, where each point represents the existence of a pair of members with a given overlay delay and a given physical delay. It may be verified that the delay between all pairs of members along the overlay is at most $160ms$, while the physical delay can be as high as $71ms$.

In future experiments, we summarize RDP results of an experiment by the *90 percentile RDP* value. We believe this is an appropriate method of summarizing results because: (i) it is an upper bound on the RDP observed by 90% of pairs of members; (ii) for pairs of members with a RDP value higher than the *90 percentile*, the overlay delay is small as discussed in the previous paragraph; and (iii) it is fairly insensitive to particular experiment parameters, unlike the omitted tail

Figure 2.15 plots the cumulative number of virtual links added and removed from the mesh as a function of simulation time. We observe that most of the changes happen within the first 4 minutes of the simulation. This is consistent with the behavior seen in Figure 3.3 and indicates that the mesh quickly stabilizes into a good structure.

Physical Link Stress: We study the variation of physical link stress under Narada and compare the results we obtain for a typical run with physical stress under DVMRP and naive unicast in Figure 2.16. One of the members is picked as source at random, and we evaluate the stress of each physical link. Here, the horizontal axis represents stress and the vertical axis represents the number of physical links with a given stress. The stress of any physical link is at most 1 for DVMRP, indicated by a solitary dot. Under both naive unicast and Narada, most links have a small stress - this is only to be expected. However, the significance lies in the tail of the plots. Under naive unicast, one link has a stress of 127 and quite a few links have a stress above 16. This is unsurprising considering that links

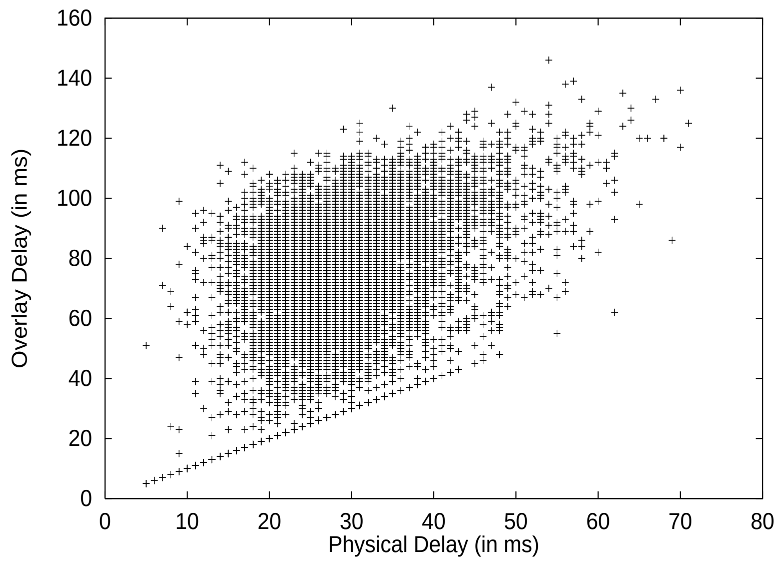

Figure 2.14: Overlay delay vs. physical delay. Each point denotes the existence of a pair of members with a given physical delay and overlay delay

near the source are likely to experience high stress. Narada however distributes the stress more evenly across physical links, and no physical link has a stress larger than 9. While this is high compared to DVMRP, it is a 14-fold improvement over naive unicast.

2.4.2 Impact of factors on performance

We are interested in studying the variation in Narada's performance due to each of the following factors: (i) topology model; (ii) topology size; (iii) group size; and (iv) fanout range. Keeping other factors fixed at the default, we study the influence of each individual factor on Narada's performance. For all results in this section, we compute each data point by conducting 25 simulation experiments and plot the mean with 95% confidence intervals.

Topology Model and Group Size: We used a Waxman topology consisting of 1024 routers and 3145 links, an ASMap topology consisting of 1024 routers and 3037 links and a Mapnet topology consisting of 1070 routers and 3170 links. We assumed a fanout range of <3-6> for all group members.

Figure 2.17 plots the variation of the 90 percentile RDP with group size for three topologies. Each curve corresponds to one topology. All the curves are close to each other indicating that the RDP is not sensitive to the choice of the topology model. For all topologies and for a group size of 128 members, the 90 percentile RDP is less than 4. For each topology, the 90 percentile RDP increases with group size. This is because an increase of group size results in an increase of mesh diameter and hence an increase of RDP.

Figure 2.18 plots the variation of worst case physical link stress against group size for three topologies. Each curve corresponds to one topology. We observe that the curves are

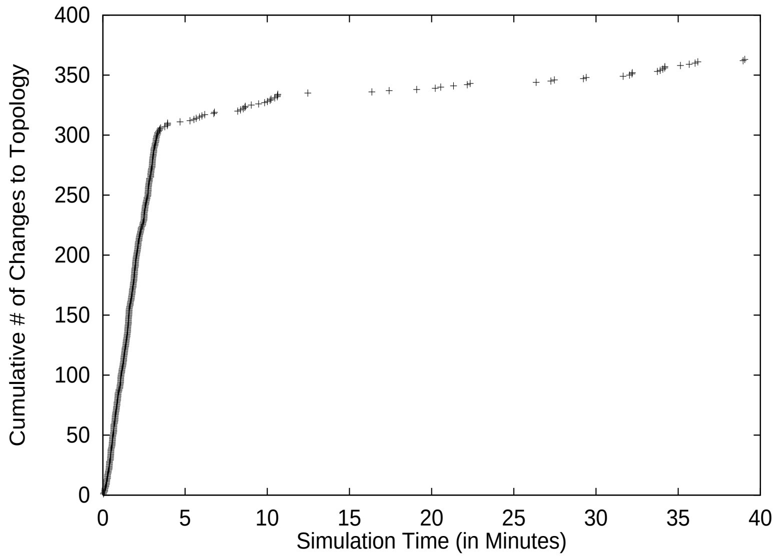

Figure 2.15: Cumulative number of virtual links added and removed vs. time

close to each other for small group sizes but seem to diverge for larger group sizes. Further, for all topologies, worst case stress increases with group size. Thus, for a group size of 64, mean worst case stress is about $5 - 7$ across the three topologies, while for a group size of 256, it is about $8 - 14$. We believe this increase of stress with group size is an artifact of the small topologies in a simulation environment relative to the actual Internet backbone. We discuss this further in Section 2.4.4.

Figure 2.19 plots the normalized resource usage (NRU) against group size for the Waxman model alone. The lower and upper curves correspond to Narada and unicast respectively. First, Narada consumes less network resources than naive unicast, and this is consistent for all group sizes. For a group size of 128, the NRU for Narada is about 1.8 and 2.2 for naive unicast. Second, NRU increases with group size. While these results imply a nearly 20% savings of network resources, we believe that the savings could be even more significant if members are clustered. We have repeated this study with the Mapnet and ASMap topologies and observe similar trends. For all topologies, the NRU is at most 1.8 for a group size of 128.

Topology Size: For each topology model, we generate topologies of sizes varying from about 64 nodes to about 1070 nodes and evaluate the impact on Narada's performance. In the experiments, we fixed the group size as 128, and a fanout range of $< 3 - 6 >$. Figure 2.20 plots the worst case physical link stress against topology size for each topology model. Across all topology models, we observe that the worst case stress increases with decrease in topology size. While the same general trend is observed for all topology models, it seems more pronounced for Waxman. We discuss this further in Section 2.4.4.

We have also studied the effect of topology size on RDP and NRU. Across all topology models, RDP appears largely unaffected by topology size, while NRU decreases with

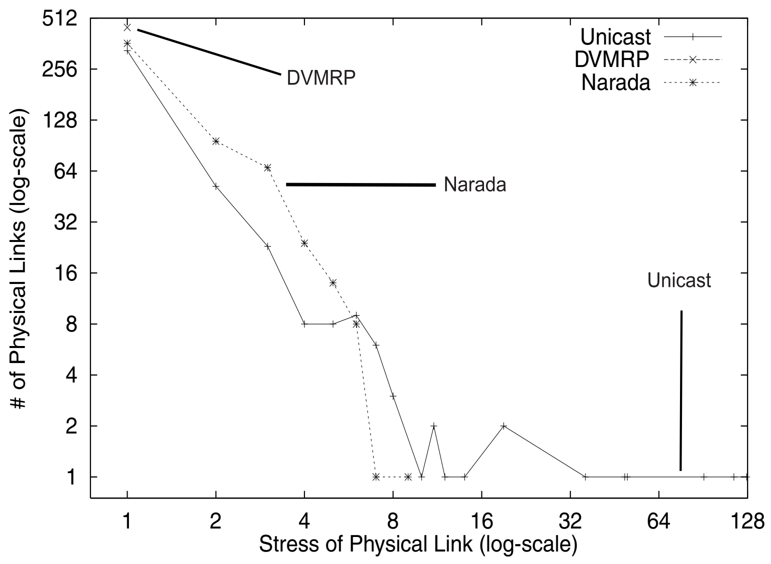

Figure 2.16: No. of physical links with a given stress vs. Stress for naive unicast, Narada and DVMRP

increase in topology size.

Fanout Range: So far, we have assumed that each member strives to maintain <3-6> neighbors in the mesh. We have investigated the effect of variation of fanout range on Narada's performance. In summary, when the fanout range increases, mesh diameter decreases and stress on links close to members increases. Consequently, RDP decreases while worst case stress increases. For a group of 128 members, as fanout range increases from <2-4> to <8-16>, the 90 percentile RDP decreases from about 5.5 to 2 while the worst case physical stress increases from about 9 to 15.

2.4.3 Protocol Overhead

Narada incurs a protocol overhead for two reasons. First, members periodically exchange routing tables and control information between each other. Second, members estimate their delays to other members by probing them periodically. We define *Protocol Overhead Ratio (POR)* as the ratio of bytes of non-data traffic that enter the network to bytes of data traffic. While we do not present results, we find that POR increases linearly with group size. Further, we note that the protocol traffic that Narada introduces is independent of source data rate and thus the POR decreases with increase in data traffic. For a group size of 128 members, the POR is about 0.25 for a source data rate of 16 kilobits per second (kbps), and less than 0.04 for a source data rate of 128 kbps. For a 64 member group and a source data rate of 128 kbps, the POR is hardly 0.02.

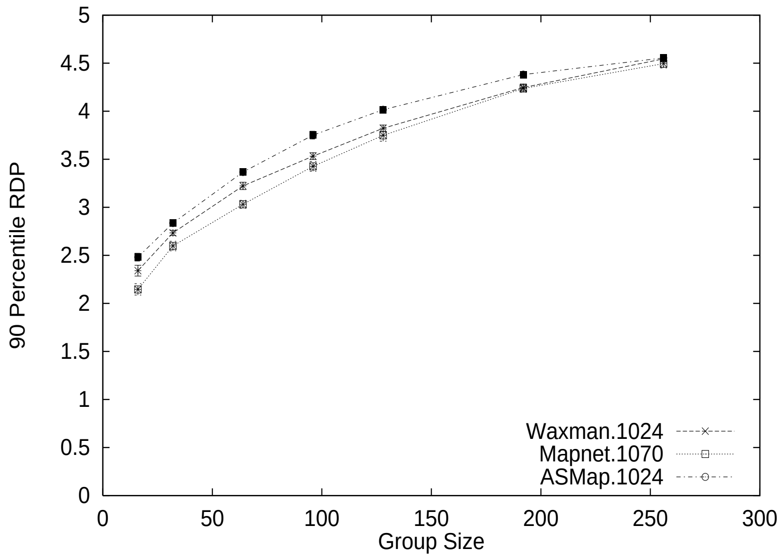

Figure 2.17: 90 percentile RDP vs. group size for topologies from three models

2.4.4 Results Summary

In this section, we summarize key results that we have presented and attempt to explain the results.

• Across a range of topology models, Narada results in a low RDP for the group sizes we consider. For a group size of 16, the 90 percentile RDP is less than 2.5, while for group sizes of 128 members, the 90 percentile RDP is less than 4. We hypothesize that RDP values might be lower on the Internet, as Internet routing is policy based and sub-optimal, while the simulator assumes shortest path routing.
• Across a range of topology models, Narada results in a low worst case stress for the group sizes we consider. For a group size of 16, the worst case stress is about 5. For a larger group size of 128 members, the worst case stress is 12, but this is still a reduction of a factor of 14 compared to unicast. We hypothesize that the worst case stress on the Internet is lower than seen in simulations. This is because the largest topologies that we use in our simulations (around 1000 nodes) are still orders of magnitude smaller than the Internet. We believe the smaller topology sizes increase the probability that an internal physical link could be shared by two overlay links that do not have any end-point in common (for example, by links $A-B$ and $C-D$, where A,B,C and D are distinct end systems). This could increase worst case stress with Narada because Narada only regulates fanout of members and does not directly regulate the stress of internal physical links.
• Narada lowers resource usage by at least 20% compared to unicast for a range of group sizes. We believe that if members are clustered, Narada can result in even larger improvement in resource usage.

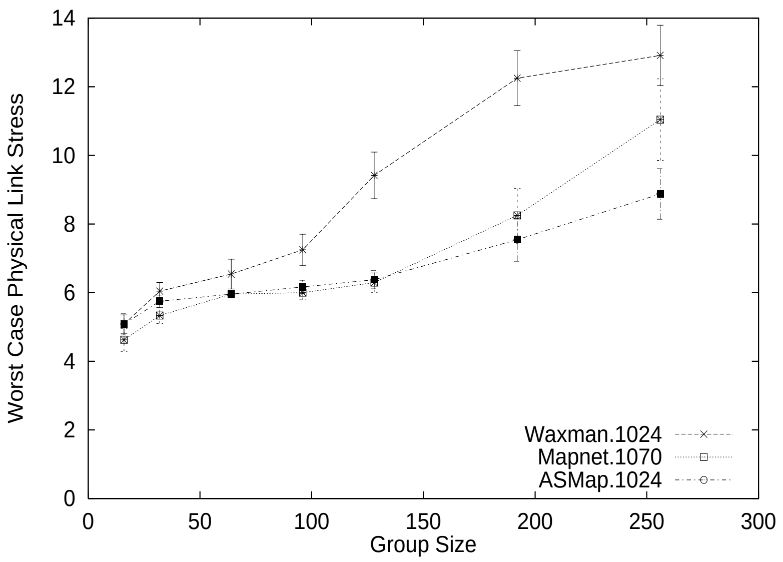

Figure 2.18: Worst case physical link stress vs. group size for topologies from three models

2.5 Summary

This chapter presents the design of Narada, perhaps the first self-organizing protocol for overlay multicast. Narada is targeted at smaller-scale, and multi-source applications. We presents the design space, key design decisions, and details of the design. We studied two fundamental performance concerns with End System Multicast in the context of Narada: increased delay penalties for receivers, and redundant copies on physical links. Our simulation results indicate that in a group of 128 members, the delay between at least 90% of pairs of members increases by a factor of at most 4 compared to the unicast delay between them. Further, no physical link carries more than 9 identical copies of a given packet. We believe these results demonstrate the viability of End System Multicast, and indicate that the performance penalties with this architecture are low.

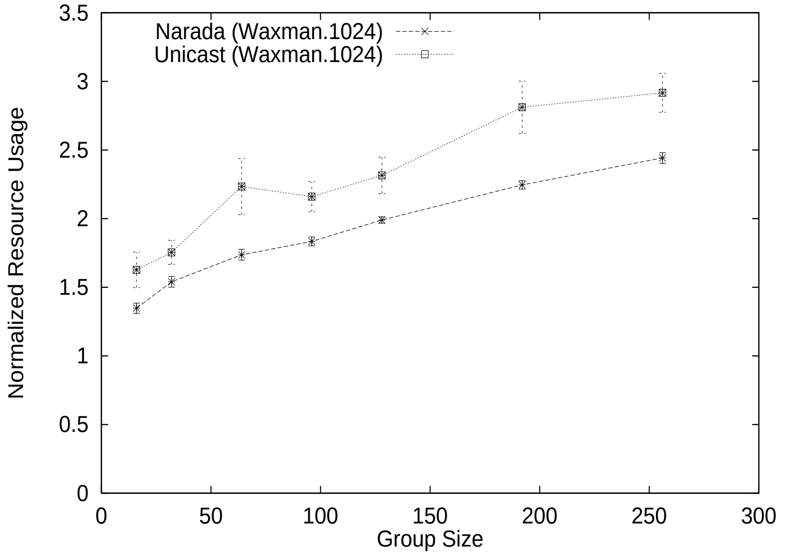

Figure 2.19: Effect of group size on NRU : Narada vs. naive unicast

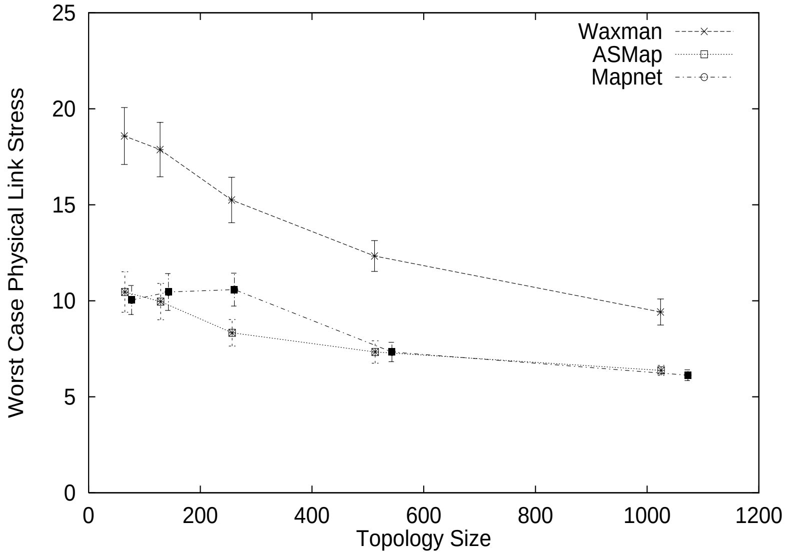

Figure 2.20: Worst case physical link stress vs. topology size for topologies from three models

Chapter 3

Optimizing overlays for bandwidth-demanding applications

Chapter 2 demonstrated that the performance penalty with End System Multicast can be acceptably low, using simulation experiments, static delay-based metrics and controlled environments. In this chapter, we evaluate the feasibility of End System Multicast to satisfy the demands of *bandwidth-demanding* applications like conferencing and broadcasting, in actual Internet environments

Our study is conducted 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 tools like *vic*[39], *vat*[32] and *rat*[27] were developed. Yet, these tools are not ubiquitously deployed today due to the limited availability of IP Multicast. Conferencing applications have stringent performance requirements, requiring not only a high sustained throughput between the source and receivers, but also require low latencies.

We show that in order to enable conferencing applications, 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 network noise and inaccuracies inherent in the measurement of these quantities. We demonstrate our ideas by incorporating them into the Narada protocol presented in Chapter 2. While we have chosen to use Narada, we believe that the techniques we present can be incorporated into other self-organizing protocols.

We evaluate our techniques by testing the 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.

The rest of the chapter is organized as follows. Section 3.1 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 3.2. Sections 3.3 and 3.4 present our evaluation methodology and results. We summarize in Section 3.6.

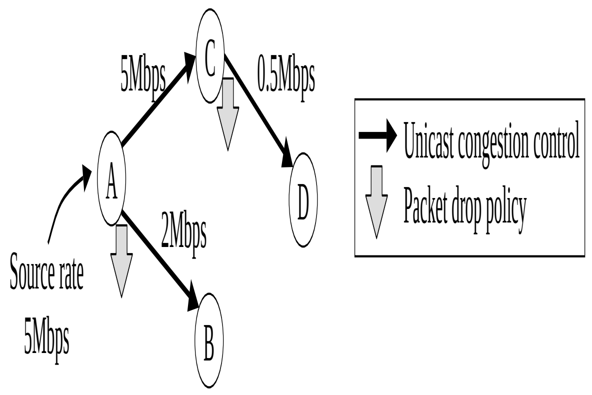

Figure 3.1: Architectural framework for supporting conferencing applications

3.1 Conferencing Applications and Overlay Design

In this section, we study the issues involved in supporting conferencing applications using End System Multicast. We begin by presenting the distinguishing characteristics of conferencing applications:

• *Performance requirements:* Conference 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:* Conference applications deal with 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.

To support conferencing applications, we consider a framework involving a hop-by-hop congestion control protocol. Congestion control on each individual overlay link is ensured by running a TCP-friendly protocol for streaming media applications [6, 21, 70]. An overlay node adapts to a bandwidth mismatch between the upstream and downstream links by dropping packets. Figure 3.1 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. Such a framework exploits the gracefully degradable nature of conferencing applications, and the application-customizable nature of the End System Multicast architecture.

We focus on addressing key performance issues with using self-organizing protocols to support conferencing applications. In particular, the issues we consider are:

• *Optimization 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.
• *Optimization 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.

We incorporate our ideas in the Narada protocol. On the one hand, Narada is explicitly designed for multi-source applications. On the other hand, the smaller group size of conferencing applications makes choosing Narada acceptable, even though Narada requires every member to maintain state about all other members.

3.2 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 demonstrate them on the Narada protocol described in Chapter 2. The key elements of Narada important for this discussion is that it constructs a mesh among participating end hosts, 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.

Constructing an overlay optimized for both latency and bandwidth presents a range of choices. In designing heuristics for tackling this problem, we have been motivated by the work done by Wang and Crowcroft [74] 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

[74]. 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 estimate 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).

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. These measurements have a low overhead. The overhead can be further reduced by querying the underlying transport protocol if this is possible.

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 not easy to obtain without active end-to-end measurements. We consider this in further detail in Section 3.5 by evaluating light-weight probing heuristics to guide parent selection.

3.3 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 overhead 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 3.3.1. Of these schemes, only one adapts dynamically to both bandwidth and latency. All other schemes consider only one of these metrics, or none at all. Section 3.4 presents detailed results that compare the performance of all the schemes. In Section 3.5, we consider light-weight probing heuristics.

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* 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.

3.3.1 Schemes for Constructing Overlays

Our schemes for constructing overlays are derived from the Narada protocol [11], 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 [11]. 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 [11]. 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.

3.3.2 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 and on different days of the week 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 group 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 3.3.1, *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.

3.3.3 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 Chapter 2 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.

We consider the *Normalized Resource Usage* of an overlay tree as defined in Chapter 2 - the ratio of the resource usage of the overlay tree 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 [15] 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, for metrics such as latency and bandwidth, we need to summarize the performance that a number of different hosts observe. Although the set of hosts and source transmission rate are identical, a particular scheme may create a different overlay layout for each experimental run. As a result, an individual host may observe different performance across the runs. However, this does not imply that the different 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. The standard deviation captures the 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. A factor that may complicate the interpretation of standard deviation is that variability in performance 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 3.4 demonstrate that some schemes are able to keep the standard deviation low. This leads us to believe the standard deviation is a reasonable measure of the variability in performance with a scheme itself.

3.3.4 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 as the underlying transport protocol on each overlay link, as discussed in Section 3.1. 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.

3.4 Experimental Results

We begin by presenting results in a typical experiment run in Section 3.4.1. Section 3.4.2 provides a detailed comparison of various schemes for constructing overlays with regard to application level performance, and Section 3.4.3 presents results related to network costs.

3.4.1 Results with a Typical Run

This section presents the performance results of our scheme in a typical experiment. It gives an idea of the performance of individual receivers throughout an experiment as a function of time. Our experiment was conducted on a week-day afternoon, using the *Primary Set* of 13 machines and at a source rate of 1.2 Mbps. The source host is at UCSB.

Figure 3.2 plots the 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 3.3 plots the RTT to a receiver, averaged across all receivers as a function of time. Again, the mean RTT to a receiver is about 100 ms on average after about 150 seconds, and remains lower than this value almost throughout the session.

Figures 3.2 and 3.3 show that in the first few minutes of the session, the overlay makes many topology changes at very frequent intervals. During this period, members are gathering network information, and improving the quality of the overlay. 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.

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 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 3.4 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

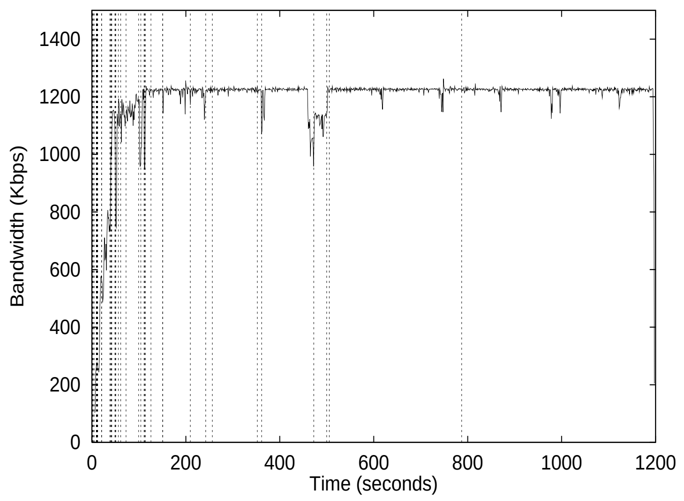

Figure 3.2: Mean Bandwidth averaged over all receivers as a function of time.

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.

3.4.2 Comparison of Schemes for Overlays

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

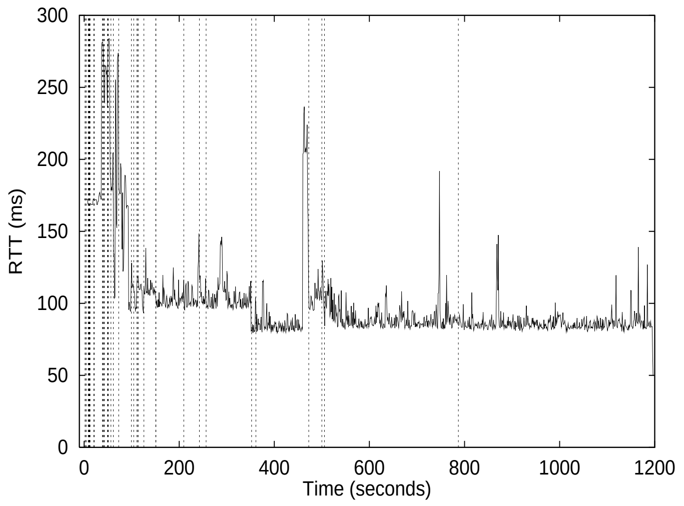

Figure 3.3: Mean RTT averaged over all receivers as a function of time.

3.4.2.1 Primary Set at 1.2 Mbps Source Rate

Figure 3.5 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, which indicates 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 machine in any run receives 1150 Kbps on average. Further, these schemes can result in consistently good performance, as seen 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 [59]. 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.

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

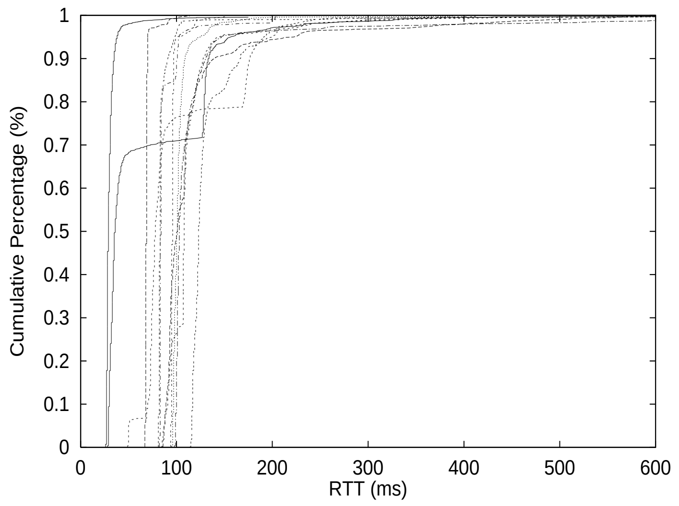

Figure 3.4: Cumulative distribution of RTT, one curve for each receiver.

Figure 3.6 plots mean RTT against rank for the same 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 perform 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.

3.4.2.2 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 3.7 and 3.8 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*

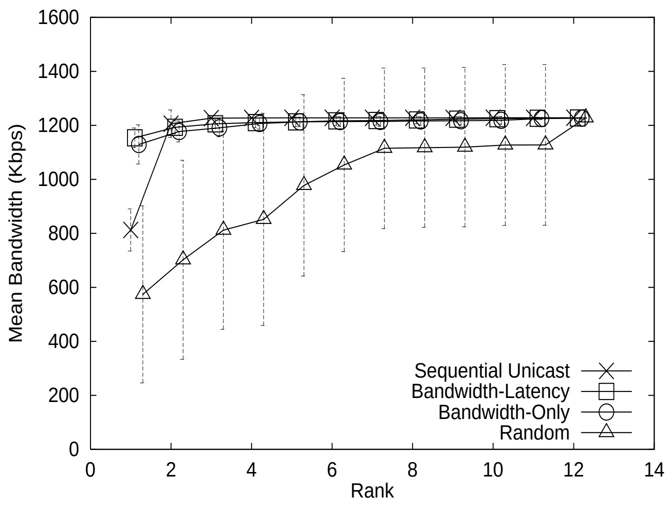

Figure 3.5: Mean bandwidth versus rank at 1.2 Mbps source rate for the *Primary Set* of machines

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.

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 [10, 24, 11], 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.

3.4.2.3 Extended Set at 2.4 Mbps Source Rate

Our results so far demonstrate that even in less heterogeneous environments such as the *Primary Set*, satisfying the requirements of conferencing applications 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

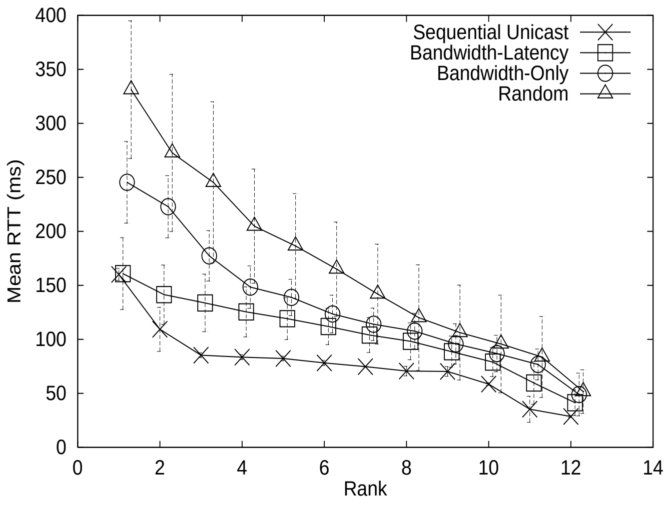

Figure 3.6: Mean RTT versus rank at 1.2 Mbps source rate for the *Primary Set* of machines

heterogeneous environments as represented by the *Extended Set*. Figures 3.9 and 3.10 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.

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 achieve RTTs more than 100 ms lower than *Bandwidth-Only*. Further, *Bandwidth-Latency* has smaller standard deviations, which indicates that the overlays produced by *Bandwidth-Latency* consistently attain good performance in both bandwidth and latency. Finally, we observe that *Bandwidth-Latency* provides almost equivalent performance to *Bandwidth-Only* in bandwidth, and to *Latency-Only* in RTT indicating that optimizing for multiple metrics does not compromise the performance with respect to any single metric.

3.4.2.4 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

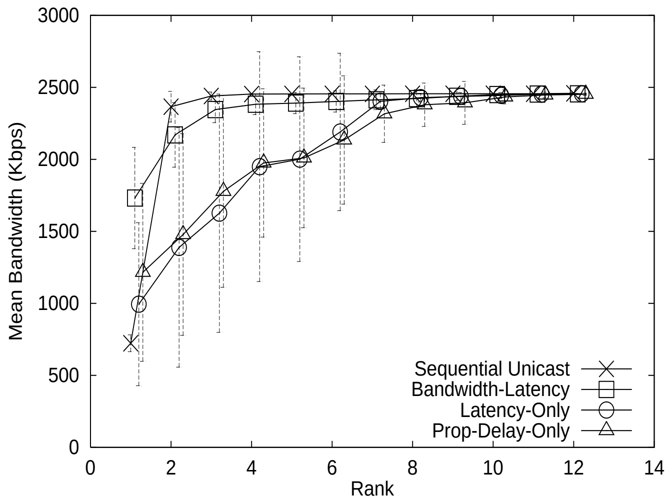

Figure 3.7: Mean bandwidth versus rank at 2.4 Mbps source rate for the *Primary Set*

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 3.1: Average normalized resource usage of different schemes

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.

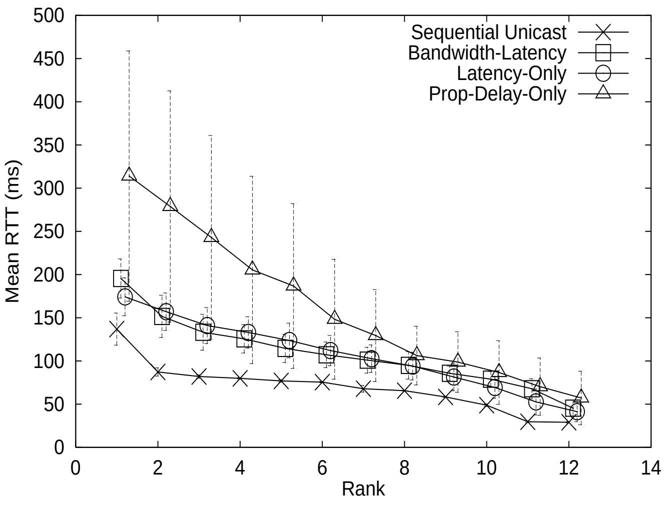

Figure 3.8: Mean RTT versus rank at 2.4 Mbps source rate for the *Primary Set*

3.4.3 Resource Usage

Table 3.1 compares the mean normalized resource usage (Section 3.3.3) 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 3.3.3. 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 stable after about four minutes.

There are several observations to be made from Table 3.1. 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. Fourth, 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. Finally, we note that *Min-Span*, the minimum spanning tree of the complete graph of all members, has a resource usage better than IP Multicast. This indicates that overlay trees can indeed have lower resource usage than IP Multicast. While minimum spanning trees are known to be optimal with respect to resource usage, it is not clear they can meet the requirements of the application.

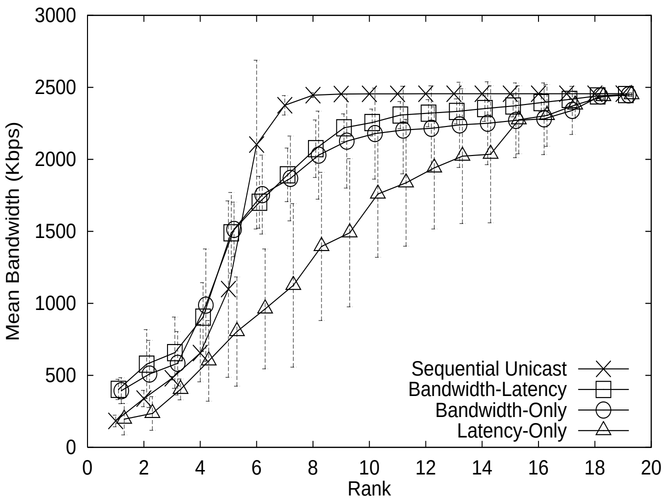

Figure 3.9: Mean bandwidth versus rank at 2.4 Mbps source rate for the *Extended Set* of machines

3.5 Evaluation of probing techniques

Our results in Section 3.4 demonstrate the importance of adapting to bandwidth. A key issue we have not discussed is choosing a parent from several prospective parents, to many of whom bandwidth is not known. Directly measuring the available bandwidth to prospective parents is costly - hence we consider light-weight probing heuristics to achieve this goal. We evaluated peer selection heuristics based on three techniques: round-trip time (RTT) probing, 10KB TCP probing, and bottleneck bandwidth (BNBW) probing.

The evaluation reported here has been conducted in conjunction with a detailed study of the performance characteristics of Napster hosts [72]. The key findings from the study relevant to this discussion are:

- All three probing techniques considered (RTT, 10KB transfer and BNBW) when used individually help select peers that provide $40 - 50\%$ of the performance with the best peer.

- The performance with the techniques can be significantly enhanced if the probing techniques are used to shortlist a small set of choices, with a oracle picking the best from among these choices In particular, choosing the best from among a set of 5 hosts shortlisted by any of the techniques produces a peer that provides more than 80% of the optimal performance.

- The three probing techniques share complementary properties and performance can potentially be improved by applying the techniques simultaneously.

These observations above motivate us to investigate heuristics that involve applying RTT-based probing heuristics to short-list a set of candidate parents, and then applying

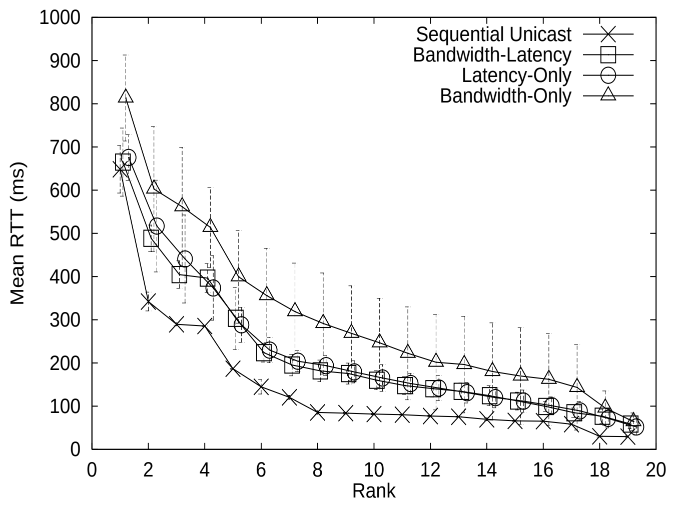

Figure 3.10: Mean RTT versus rank at 2.4 Mbps source rate for the *Extended Set* of machines

other light-weight probing techniques to make the final choice. We chose RTT-based probing heuristics to produce a short-list (of five hosts) in the first step because of its very low overhead and light-weight nature.

We considered the following techniques:

- *Random*: This technique chooses a peer at random.
- *RTT*: This technique uses single-packet RTT probe to select peers. In our experiment, we conduct RTT probes in parallel to all peers of interest, and select the one with the smallest RTT value.
- *RTT filter + 10K*: This technique selects at most 5 of the candidate peers based on the best RTT estimates and perform 10KB of file transfer using TCP in parallel. The peer with the shortest transfer time is selected. Use of 10KB probes has been suggested in Overcast [34].
- *RTT filter + 1-bit BNBW*: This technique selects at most 5 peers with the lowest RTT estimates and then chooses the peer with the highest bottleneck bandwidth. Ideally we would like to measure the bottleneck bandwidth between peers during our experiments. However, the bottleneck bandwidth measurement tools we are aware of all require super-user access, which is a privilege we do not have on the majority of our testbed machines. Hence, we decide to use only a single bit to differentiate between ADSL and non-ADSL peers and study how much can such minimal information help.

To select a good performance metric to evaluate the probing techniques for End System Multicast, we need to consider some of its unique characteristics. First, a peer is satisfied with a parent peer as long as it can receive data at the source rate. The maximal achievable throughput is not important. Second, as we show later in this section, due to the continuously adaptive nature of overlay multicast streaming, given sufficient time, the overlay

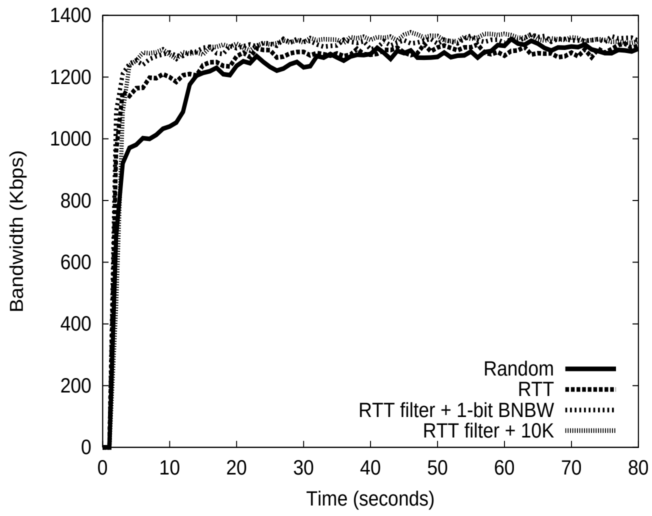

Figure 3.11: Mean receiver bandwidth as a function of time at 1.5 Mbps source rate. Only the first 80 seconds of the experiments are shown here.

structure will converge to a stable configuration, where each peer is receiving data at as high a quality as possible. Thus, the key performance metric is the *convergence time.*

We define the convergence time of a peer to be the amount of time after the initial join it takes for the peer to receive more than 95% of the *stable bandwidth* for 30 seconds. We determine the stable bandwidth of a peer based on the bandwidth it receives at the end of a 5-minute experiment. The 30 second window is necessary because a peer's performance may momentarily dip below 95% of the stable bandwidth due to network performance fluctuations.

We evaluate the probing techniques on the Internet test-bed using the *Extended Set.* Every individual experiment is conducted in the following fashion. Members join the group in a sequential order, spaced by 2 seconds apart. The order in which members join is randomized in each experiment. The source multicasts data at a constant rate of 1.5Mbps. Each experiment lasts for 5 minutes.

Figure 3.11 plots the mean bandwidth, averaged across all receivers and multiple experiments as a function of time. Each curve corresponds to one probing technique. For clarity, only the result of the first 80 seconds of the experiments are shown. For all techniques, the overlay reaches the same stable performance within 30 seconds. However, random peer selection leads to a longer time to converge compared to techniques that exploit network information. These results show that while a self-improving overlay eventually converges to stable performance, a better peer selection technique can help to improve overlay convergence.

Figure 3.12 shows the cumulative distribution of convergence time of different individual receivers. First, regardless of the peer selection technique used, almost all peers converge to receive good bandwidth within 60 seconds. Second, *Random* results in longer convergence time with as many as 30% of the peers taking longer than 15 seconds to converge. The use

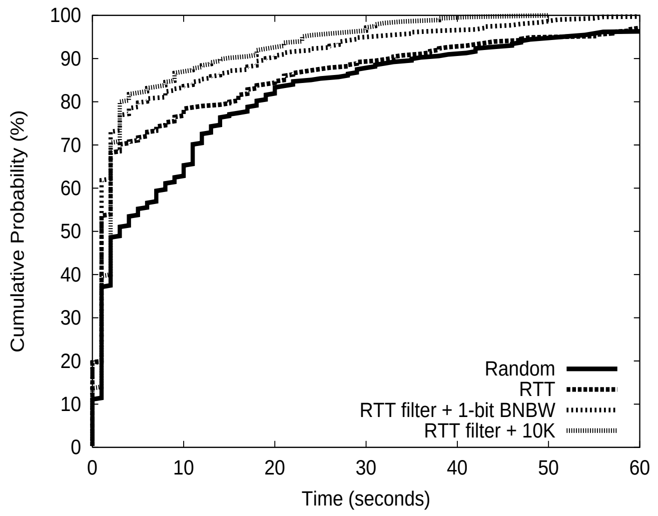

Figure 3.12: Cumulative distribution of convergence time for basic techniques.

of a simple RTT technique greatly improves convergence time, and 70% of the peers take less than 5 seconds to converge. This is an indication that RTT can successfully help peers to select good parent peers when they first joined the group. Finally, RTT filter with either 10K or 1-bit BNBW results in better convergence property than RTT alone, and 80% of the peers take less than 5 seconds to converge. Note that even though our low-fidelity 1-bit BNBW metric can only differentiate the 3 ADSL host from the rest, it is shown to have benefits.

Our results so far indicate that simple light-weight techniques such as RTT filter with 1-bit BNBW can achieve good convergence properties compared to random peer selection. We have considered whether more sophisticated ways of combining these techniques can help to further improve performance. Our results indicated that there is not much benefit to this, and we refer the reader to [72] for further details.

3.6 Summary

In this chapter, we considered the viability of End System Multicast to supporting bandwidth-demanding applications in real Internet settings. We focused our study on conferencing applications, that simultaneously need high bandwidth and low latencies. We considered how the Narada protocol may be customized for conferencing applications, and presented techniques that enable Narada to adapt to both bandwidth and latency in the overlay construction.

A detailed evaluation on a real-world Internet test-bed demonstrates that our heuristics work well. 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 also demonstrate the importance of adapting to 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.

Finally, we evaluated light-weight probing heuristics that can guide parent selection while constructing overlays, given that probing nodes to determine their bandwidth can have a high overhead. Our evaluation show that RTT-based probing techniques are effective in reducing convergence time. While combining RTT-based techniques with 10-Kbyte bandwidth probes and 1-bit bottleneck bandwidth information helps, the improvement in performance is not significant.

Overall, our results show the promise of enabling bandwidth-demanding applications on the Internet using End System Multicast.

Chapter 4

Experience Building and Deploying an ESM-Based Broadcasting System

Chapter 2 and Chapter 3 evaluated the performance of End System Multicast prototypes using simulation studies and experiments on wide-area Internet test-beds. In this chapter, we take these ideas further, and present our experience building and deploying a fully operational broadcasting system based on End System Multicast. On the one hand, deployment of applications enhances the credibility of End System Multicast, and is perhaps the only way to eventually validate the architecture. On the other hand, experience gained from such usage guides us to key challenges that future iterations of the system design must focus on.

While the earlier chapters in this dissertation focused on conferencing applications, the system targets broadcasting applications. Like conferencing applications, broadcasting applications are bandwidth-demanding and involve video bit rates of several hundred kilobits per second. In addition, both applications involve gracefully degradable media streams and can tolerate losses by degrading stream quality. However, while conferencing applications involve multiple sources and smaller group sizes, broadcasting applications typically involve a single source, and much larger group sizes. Further, while conferencing applications are interactive and require latencies of the order of a few hundred milliseconds, broadcasting applications tend to be live rather than interactive, and can tolerate latencies of several seconds.

The system has been built in the context of the End System Multicast project at Carnegie Mellon. In building the system, we have adopted simple or natural solutions, with the provision that the design decisions could be revisited in the light of future experience. This approach has accelerated the deployment of the system, and, consequently has led to faster feedback from real deployment.

This thesis has led the design and implementation of *Sparta,* the self-organizing protocol in the system. The design of Sparta is *integrated* with the overall objectives of validating the ESM architecture, and building an operational system. Sparta builds on our prior experience with Narada. However, key differences arise in the group management algorithms used in the protocol, given that Narada was targeted at smaller-scale multi-source conferencing applications, while Sparta is targeted at larger-scale single-source broadcasting applications. Further, implementing the protocol in the context of a real system has led to several refinements of the ideas in Narada, and this has led to a unique implementation-oriented

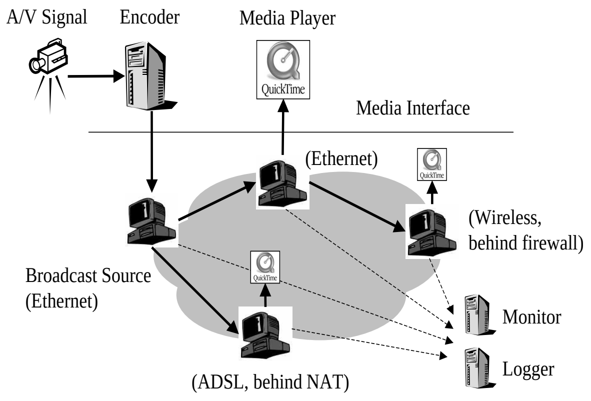

Figure 4.1: Broadcast system overview.

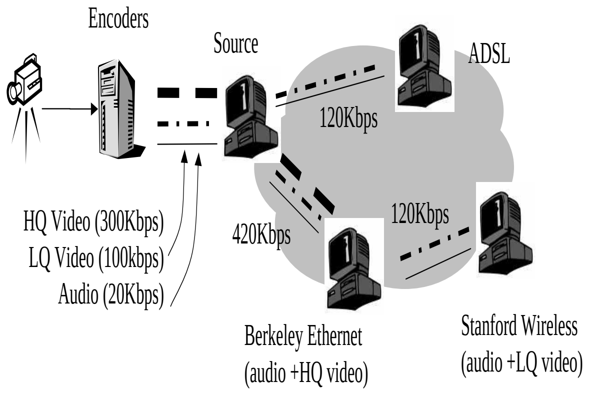

Figure 4.2: Single overlay approach to host heterogeneity.

perspective to protocol design.

In over a year, the system has been providing a cost-effective alternative for Internet broadcasts. It has been used to broadcast tens of events, and has been used by thousands of users spread across multiple continents in home, academic and commercial environments. Technical conferences and special interest groups are the early adopters. Our experience confirms that End System Multicast can be easily deployed and can provide good application performance. The experience has also highlighted key issues that future designs of the system must address, and we believe are of broader interest to the community.

The rest of the chapter is organized as follows. Section 4.1 provides a summary of the ESM broadcasting system, the detailed description of which can be found in [12]. Section 4.2 provides a detailed description of the self-organizing protocol used in the system. We discuss the deployment of the system, and the performance we have seen in Sections 4.4 and 4.5. We present some of these key lessons in Section 4.6.

4.1 ESM Broadcasting System

Figure 4.1 gives a high-level overview of our broadcasting system. The encoder takes the multimedia signal from the camera, converts into audio and video streams, and sends to the broadcast source. The broadcast source and receivers run an overlay multicast protocol to disseminate the streams along the overlay. Each receiver gets the broadcast stream, and forwards to the media player running on the same machine. In addition, the participating hosts send performance statistics to the monitor and log server for both on-line and post-mortem analyses.
The system has three main components:

- *Overlay protocol for data dissemination:* This is the protocol used to construct efficient overlay trees among participating hosts. We present this in greater detail in Section 4.2.

- *Support for receivers with heterogeneous capabilities:* Internet hosts are heterogeneous and may have a wide range of receiving bandwidth capabilities. The system tackles this by encoding video at multiple bit-rates in parallel and broadcasting them simultaneously, along with the audio stream, through the overlay as shown in Figure 4.2. Further, a unicast congestion control protocol is run on the data path between every parent and child, and a prioritized packet forwarding scheme is employed at each overlay link. Audio is prioritized over video streams, and lower quality video is prioritized over higher quality video. The system dynamically selects the best video stream based on loss rate to display to the user. When a receiver does not have sufficient bandwidth to view the high quality video stream, or when there are transient dips in available bandwidth due to congestion, as long as the lower quality video stream is received, a legible image can still be displayed. This design can be seamlessly integrated with layered codecs if available. We used TCP as the unicast congestion control protocol, but more recent deployments use TFRC [21], a UDP-based congestion control protocol.

- *Support for receivers behind NATs and firewalls:* Hosts behind NATs cannot initiate communication to other hosts behind NATs. The system employs heuristics that enable hosts behind NATs to be parents of hosts with public IP addresses, (though they cannot be parents of other hosts behind NATs). The reader is refered to [12] for further details.

We use *QuickTime* [47] as the media player. This choice was motivated by the fact that it is widely available and runs on multiple popular platforms. We use *Sorenson 3* [65] and MPEG4, both of which are supported by QuickTime, as video codecs. Typical bit rates used in our broadcasts are audio at 20 Kbps, and two target video bit-rates of 100 kbps and 300 kbps.

The broadcasting system is supported in Linux, Windows and MAC. Receivers tune into the broadcast using a simple web-link interface. The system includes a publishing toolkit [18] which allows content publishers to easily set up a broadcast of their own, and a monitoring system which provides content publishers with online information about individual participating hosts, the current overlay tree, the bandwidth on each overlay link, and the current group membership.

4.2 Sparta Description

In this section, we describe the overall goals and approach while designing Sparta, key differences from the Narada protocol presented in earlier chapters, and present a detailed description of the protocol implementation.

4.2.1 Design Approach

The decisions taken while designing and implementing Sparta are best understood by appreciating the broader context in which the effort was undertaken. The design of Sparta is integrated with the overall objectives of validating the ESM architecture, and building an operational system. Putting together an operational system involves making design decisions on several individual components of the system. The trade-offs between the design choices, and the significance of the design choice in the overall context of the system performance have not always been apparent. Our overall approach has been to emphasize putting together a complete prototype to help us appreciate the range of issues involved, while adopting simple or natural design choices for any given component. Our belief has been that the overall experience could guide us to key design decisions that need to revisited and more carefully examined, which could in turn lead to refinements in the next iteration of the system design.

4.2.2 Overview

Sparta is distributed, self-organizing and performance-aware, and constructs a tree rooted at the source. The tree is optimized primarily for bandwidth, and secondarily for delay. We use a distributed protocol, as opposed to a centralized protocol to minimize the control overhead at the source.

While we have adopted a tree-based data delivery structure, performance can potentially be improved by constructing more resilient data delivery structures. In particular, recent work has investigated mechanisms for redundancy in conjunction with specialized coding algorithms [43, 8]. While we hope to leverage these ideas in future designs, our design has been influenced by practical system constraints on an immediately deployable operational system, and our desire to interoperate with commercial media players and a wide range of popular codecs.

Contrast with Narada: Sparta builds on our experience with Narada. However, key differences arise from the fact that Sparta is targeted at single-source broadcasting applications, while Narada targeted multi-source conferencing applications. The goal of supporting multiple sources motivated the design of a two step process for tree construction in Narada, with the first step involving the construction of a mesh, and the second step involving the construction of spanning trees of the mesh using routing algorithms. In contrast, given that the system deals with single source applications, the new protocol adopts a direct single step process for tree construction, with members directly choosing parents from other members they know.

The Narada protocol required members to maintain knowledge of all other group members. This design point was chosen given that the design targeted conferencing applications that typically involved smaller group sizes. However, the control overhead becomes significant for much larger group sizes that the broadcasting system is targeted for. This leads us to adopt more scalable group management algorithms in Sparta, where each member knows only a small subset of the group members.

4.2.3 Protocol Description

We present the various components of Sparta, and the details of each component.

4.2.3.1 Group Management

The group management algorithms refer to the mechanisms that enable each member to maintain knowledge of only a small subset of group members, irrespective of the total number of members in the group. In our protocol, each member maintains information about a random subset of members not correlated to the tree used for data delivery. Each member also maintains information about the path from the source to itself, through periodic control packets exchanges between every parent and child in the tree. A new host joins the broadcast by contacting the source and retrieving a random list of hosts that are currently in the group, based on the knowledge with the source. It then selects one of these members as its parent using the parent selection algorithm (Section 4.2.3.4). A member that is disconnected from, or is not satisfied with the performance from its parent chooses a new parent from among the members it knows using the same parent selection algorithm.

Members maintain a random list of group members by using a gossip protocol adapted from [52]. Each host A periodically picks a member (say B) at random, and sends B a subset of group members that A knows, along with the last timestamp it has heard for each member. When B receives a membership message, it updates its list of known members by creating entries for members it learns about newly, and by refreshing the timestamp for members it already knows about. Each members delete state of other members if the state has not been refreshed in a certain period (*timeout*)

The overhead due to messages associated with the group management algorithm depends on the time between consecutive gossip messages sent by a node, and the number of members about which information is sent in each gossip. To keep the overhead small, we assume gossip messages are sent every two seconds and carry information about eight members.

In the algorithm above, membership information may be stale - that is, entries in a member's knowledge table may correspond to members that have left the group. The choice of timeout value, *timeout*, is important. Choosing a high value could result in a significant fraction of member entries corresponding to dead nodes. However, making it too small could be too aggressive, and restricts the total number of nodes that a host knows. In our implementation, we use a timeout value of 5 minutes.

Design Discussion: Our group management algorithm constructs a knowledge graph where each member knows a random subset of members in the group. One design alternative is to organize members into a more hierarchical structure. Our current design choice has primarily been motivated by its simplicity. While it is possible that hierarchical approaches may have better scaling properties, they involve a higher implementation complexity, and our preliminary investigations did not reveal a clear performance benefit. We defer a detailed investigation of the trade-offs between the design choices for group management algorithms to future work.

There are two dimensions in we hope to refine the current group management algorithms in the future at much larger scales. First, the current mechanisms are not explicitly optimized for clustering receivers at nearby locations. A possible improvement is to have the source provide more informed group membership subsets to nodes, perhaps by leveraging knowledge of their network coordinates [71]. Second, in our group management algorithm, all nodes that join contact the source to get boot-strapped. This can potentially be an issue at much larger scales in the presence of flash-crowds, or many simultaneous joins in the system. We hope to investigate these concerns and incorporate mechanisms to tackle the issues in future designs.

4.2.3.2 Handling Group Membership Dynamics

Dealing with graceful member leave is fairly straight-forward: hosts continue forwarding data for a short period, while its children look for new parents using the parent selection method described below. This serves to minimize disruptions to the overlay. Hosts also send periodic control packets to their children to indicate live-ness. If a child does not receive control packets from the parent for a certain period of time, it assumes the parent is dead, and finds a new parent using the parent selection algorithm described later. The detection time, or time taken to detect death is set at about 5 seconds, as explained in the next subsection.

4.2.3.3 Performance-Aware Adaptation

We consider three dynamic network metrics: available bandwidth, latency and loss. There are two main components to this adaptation process: (i) detecting poor performance from the current parent, or identifying that a host must switch parents, and (ii) choosing a new parent, which is discussed in the *parent selection* algorithm.

Detection Time: Each host maintains the application-level throughput it is receiving in a recent time window. If its performance is significantly below the source rate (less than 90% in our implementation), then it enters the probe phase to select a new parent. One of the parameters that we have found important is the *detection time* parameter, which indicates how long a host must stay with a poor performing parent before it switches to another parent. We employed a detection time of 5 seconds. The choice of this timeout

value has been influenced by the fact that we are running a congestion control protocol on the data path (TCP or TFRC). Switching to a new parent requires going through a slow-start phase, and we have observed it may take 1 - 2 seconds to get the full source rate.
Constrained hosts: Our experience reveals the need for the protocol to adaptively tune the detection time because: (a) many hosts are not capable of receiving the full source rate, (b) even hosts that normally perform well may experience intermittent local network congestion, resulting in poor performance for any choice of parent, (c) there can be few good and available parent choices in the system. Changing parents under these environments may not be fruitful. We have implemented a simple heuristic for dynamically adjusting the detection time, involving an increase if several parent changes have been made recently, and a decrease if it has been a long time since the last parent change.
VBR Streams: Our initial implementation did not consider loss rate as a metric. We found it necessary to consider loss-rates while dealing with variable-bit-rate streams. For example, in one of our broadcasts, the source rate was 429 kbps on average, with a standard deviation of 65 kbps, and a peak of 3 Mbps. If loss rates were not considered, dips in the source rate would cause receivers to falsely assume a dip in performance and react unnecessarily. Our solution avoids parent changes if no packet losses are observed despite the bandwidth performance being poor.
Adaptation Policy: The failure of a node or congestion on an overlay link can impact a large number of down-stream recepients. Ideally, we would like the reaction to these events to be coordinated, with only the immediately affected children switching parents. However, there are two difficulties. First, there is an implementation complexity associated with achieving such coordination in a distributed fashion. Second, it is possible that the root node of the affected sub-tree may be unable to fix the problem because of congestion close to it, while its descendants have the potential to. Our current implementation chooses a solution with minimal coordination: all end systems observe their current performance, and independently switch parents if the performance is not good for a detection time window. In the case a host gracefully leaves, given that it informs its children first, it is likely the immediate children alone will react and fix the problem. With abrupt death of members, or network congestion however, it is likely that all descendants in a sub-tree detect such an event simultaneously, and independently fix the problem. Our design choice has been motivated by its simplicity - we defer to future work an investigation of the trade-offs between the performance benefits of more coordinated schemes, and the implementation complexity associated with them.

4.2.3.4 Parent Selection

When a host (say A) joins the broadcast, or needs to make a parent change, it probes a random subset of hosts it knows. The probing is biased toward members that have not been probed or have low delay. The number of people probed in each cycle must be large enough that a reasonable selection of parent can be found, but not too large that this becomes a high overhead. Our current implementation involves probing about thirty hosts.

Some of the hosts that A probes may no longer be in the group given the group management algorithm can involve stale entries. Each host B that is still in the group responds to the probe providing information about: (i) the performance (application throughput in the recent 5 seconds, and delay) it is receiving; (ii) whether it is degree-saturated or not; and (iii) whether it is a descendant of A. The probe also enables A to determine the round-trip time to B.

A waits for responses for a timeout period of 1 second. The choice of the timeout period was motivated by it being a large enough value of Internet round-trip times that maximizes the number of responses received from members. From the responses A receives, it eliminates those members that are saturated, or who are its descendant. From the remaining members, a node is picked as parent as described in the next paragraph. While eliminating descendants helps avoid loops, such information may be inconsistent. In such cases, a parent change by A may result in a loop. This is detected after the fact through the normal path update operations, and all affected members employ a fresh round of parent selection. From our experience, we have found the optimistic strategy to work reasonably well, and loop occurence is rare.

In order to assess the number of children a parent can support, we ask the user to choose whether there is at least a 10 Mbps up-link to the Internet. If so, we assign such hosts a degree bound of 6, to support up to that many number of children. Otherwise, we assign a degree bound of 0 so that the host does not support any children. We hope to incorporate mechanisms that can automatically detect the access bandwidth of the host in the future.

For each member B that has not been eliminated, A evaluates the performance (throughput and delay) it expects to receive if B were chosen as a parent. The expected delay is the sum of the delay B is experiencing from the source, and the delay of the overlay link $B - A$ estimated from the probe response of B. A computes the expected application throughput as the minimum of the throughput B is currently seeing and the available bandwidth of the path between B and A.

A shortlists B for further consideration if either: (i) switching to B has the potential to substantially improve application throughput; or (ii) switching to B provides as much or better throughput while substantially improving delay. B substantially improves throughput to A if picking B as parent can result in the delivery of a higher quality of video than that currently being received. B substantially improves delay to A if the fractional decrease in delay exceeds a threshold (25% in our implementation). From among the shortlisted candidates, A picks the parent that can offer the best bandwidth, and breaks a tie by picking a parent that has the lowest delay to A. If no node is shortlisted, then A continues with its current choice of parent.

One difficulty in the heuristics above is that they assume A knows the available bandwidth of the link $B - A$. However, in practice, the available bandwidth of paths cannot be measured unless there is data flow along the path. The protocol tackles this by maintaining a history of performance of past parents - if A has previously chosen B as parent, then it has an estimate of the bandwidth of the overlay link $B - A$. If there are some hosts to which bandwidth is known (likely because they have been chosen in the past), then A picks one of them as parent if they are known to provide good bandwidth. If they are known to not provide good bandwidth, or the bandwidth to hosts is not known, then A picks a parent based on delay, based on our results in Chapter 3 which show the promise of light-weight

RTT-probing heuristics.

4.3 Evaluation Overview

In the rest of this chapter, we present our experience with the ESM-based broadcasting system. We organize our discussions by focusing on a few key questions:

• How easy is to deploy applications using End System Multicast?
• How well does the system perform in terms of giving good performance to the user? How does the environment affect system performance?
• What design refinements could have led to better performance, and have been highlighted from our experience?

Section 4.4 presents a detailed description of the deployment we have achieved with our system. We describe our deployment both in terms of the usage of our system, and the diversity of deployment achieved. We also characterize the environments we see in our broadcasts, as these characteristics play a key role in impacting the performance of the system. We present results from broadcasts in Section 4.5, analyzing the system from a variety of perspectives. Finally, in Section 4.6, we quantitatively analyze the performance benefits that may accrue from key design modifications motivated by our experience.

4.4 Deployment

In this section, we present the deployment status of the ESM Broadcasting System. Our goal is to both present an idea of how wide and successful our deployment is, as well as to characterize key charcteristics of the environments of our broadcast that may affect the performance results.

Over the last year, the system has been used by at least 4 content publishers besides ourselves to broadcast tens of events, has accumulated hundreds of operational hours, and has been used by several thousand participants. Most broadcasts involved technical content like workshops and conferences, with audience tuning in because they were interested in the content, but could not attend the events in person. We also conducted an experimental broadcast to the Slashdot [63] community (a web-based discussion forum). In this broadcast, some of the audience tuned in for the content, while others tuned in because they were curious about the system.

The ultimate goal of ESM is to investigate purely application-end point architectures, where overlays are constructed only among group members actually participating in the broadcast. However, the viability of the architecture depends on the ability of participating hosts to support other children, which depends both on the bandwidth resources of participating hosts, and the willingness of users to contribute their bandwidth resources. Our deployment efforts were motivated by the objectives of investigating how far we could go with purely application end-point architectures, and to gain more experience with issues such as the availability of bandwidth resources in real deployment scenarios. However unlike conventional research experiments, our deployment needs to run in real-life, involves convincing event-organizers to use our system, and involves real users who primarily care about the final content delivered. Failures would damage our credibility, and limit future adoption of our system.

Given these constraints, our deployment employed PlanetLab [46] machines, which we call waypoints, to join the broadcast, in addition to the real participants. From the perspective of the system, waypoints are the same as normal participating hosts and run the same protocol – the only purpose they served was increasing the amount of resources in the system. To see this, consider Figure 4.3, which plots a snapshot of the overlay during the SIGCOMM2002 broadcast. The shape and shade of each node represents the geographical location of the host as indicated by the legend. Nodes with a dark outer circle represent waypoints. We see that waypoints are scattered around at interior nodes in the overlay, and may have used normal hosts as parents. Thus they behave like any other user, rather than statically provisioned infrastructure nodes. While our use of waypoints so far has prevented direct conclusions about purely application end-point architectures, our subsequent analysis in Section 4.6 studies the role played by waypoints in the broadcast, and points to opportunities for the reduced use of waypoints in subsequent broadcasts.

Table 4.1 lists the major broadcasts, duration, number of unique participants, number of waypoints used in each broadcast and the peak group size. The broadcast events attracted from 15 to 1600 unique participants throughout the duration. The peak sizes, or number of simultaneous participants in any broadcast ranged from about 10 to 280. Efforts are ongoing to attract larger peak sizes.

Table 4.3 presents detailed information regarding user dynamics and composition of hosts in our broadcasts. Both factors influence the performance of a self-organizing protocol. The more dynamic an environment, the more frequently a host is triggered to react. The more unfavorable the composition of hosts, the longer it could potentially take to discover a good parent. We present results from 6 of our larger broadcasts, 5 of which were conference/lecture-type broadcasts, and the other being *Slashdot*. For multi-day events, such as SIGCOMM2002 and SIGCOMM2003, we present results from one day in the broadcast. For Slashdot, we present results for the first 8 hours. We define an *entity* as a unique user identified by its $< publicIP, privateIP >$ pair. An entity may join the broadcast many times, perhaps to tune in to distinct portions of the broadcast, and have many *incarnations*. Our analysis reports on incarnations unless otherwise stated.

Table 4.3 lists the mean session interarrival time in seconds for the 6 broadcasts in the fourth column. The Slashdot broadcast has the highest rate of group dynamics compared to all other broadcasts using our system. For the five broadcasts of conferences and lectures, the mean interarrival time was a minute or more, whereas the interarrival time for Slashdot was just 17 seconds.

Table 4.3 also presents two different measures of session duration: individual incarnation duration and entity duration (cumulative over all incarnations) which captures the entity's entire attention span. For entity session duration, again, we find that all 5 real broadcasts of conferences and lectures have a mean of 26 minutes or more, and a median of 16 minutes or more. In the SIGCOMM2002 broadcast, the median was 1.5 hours which corresponds to one technical session in the conference. To contrast, the Slashdot audience has a very short attention span of 11 and 7 minutes for the mean and median. This indicates that the Slashdot audience may have been less interested in the content. The incarnation session duration also follows a similar trend with shorter durations. We note that the shorter incarnation session durations with SIGCOMM2003 and Lecture1 were caused by a couple of entities testing the system and joining and leaving frequently.

Table 4.3 presents information regarding the composition of participating hosts in the system, by presenting the the percentage of incarnations in the system that were eligible as parents (based on whether their out-going bandwidth was sufficient). The table also presents the percentage of incarnations which are public hosts, and which can serve as parents. The 5 conference and lecture broadcasts have the same trend, with 44% or more incarnations that can serve as parents. On the other hand, only 19% of incarnations could be parents in Slashdot. Further, when we consider the fraction of *public* hosts that could be parents, we find this ranges from $17 - 57\%$ for the conference-style broadcasts, but is just 7% for the Slashdot broadcast. This indicates that there were much less available resources in the system in the Slashdot broadcast.

Table 4.2 presents a more detailed view of the diversity of hosts that took part in two of the large broadcasts (SIGCOMM2002 and Slashdot). The deployment has reached a wide portion of the Internet - users across multiple continents, in home, academic and commercial environments, and behind various access technologies. Further, we see that in the Slashdot broadcast over 66% of the hosts are behind cable/DSL and cannot take children, while 68% of the hosts are behind NAT, and can only support public hosts as children. This is in contrast to the SIGCOMM2002 broadcast where only 20% of the hosts are behind cable/DSL. Note that we did not have NAT/firewall support in the SIGCOMM2002 broadcast.

Summary: We have been successful in achieving wide-scale deployment with our ESM-based broadcasting system. The number of users and the diversity of users demonstrates the deployment potential of End System Multicast. We wish to contrast this to the MBone [7], the usage of which was primarily restricted to researchers in academic institutions. We have also characterized key aspects of our broadcast. Most of our broadcasts have been conference/lecture-style, with inter-arrival times of the order of a few minutes, incarnation stay-time durations of around ten minutes, and a large fraction of incarnations being eligible as parents. The Slashdot broadcast on the other hand had inter-arrival times of 17 seconds, incarnation stay-time of 3 minutes, and with only 7% of the total incarnations being eligible as parents.

4.5 Performance Results

In this section, we present results evaluating the system from several perspectives. Section 4.5.1 presents the average performance of users, measured in terms of the mean application-level throughput a user sees during the session, and subjective feedback provided by users. Section 4.5.2 present results that correlate performance users see with how long they have stayed in the group. Section 4.5.3 provides more detailed results on the transient performance of users providing performance of audio and different video quality

Event	Duration (hours)	Unique Hosts/ Waypoints	Peak Size/ Waypoints
SIGCOMM2002	25	338/16	83/16
SIGCOMM2003	72	705/61	101/61
DISC2003	16	30/10	20/10
SOSP2003	24	401/10	56/10
Slashdot	24	1609/29	160/19
DARPA Grand Challenge	4	800/15	280/15
Distinguished Lectures Series (8 distinct events)	9	358/139	80/59
Sporting Event	24	85/22	44/22
Commencement (3 distinct events)	5	21/3	8/3
Special Interest	14	43/3	14/3
Meeting	5	15/2	10/2

Table 4.1: Summary of major broadcasts using the system. The first 4 events are names of technical conferences.

	SIGCOMM2002 broadcast 8/2002 9am-5pm (total 141 hosts)				
Region	North America (101)	Europe (20)	Oceania (1)	Asia (12)	Unknown (7)
Background	Home (26)	University (87)	Industry (5)	Government (9)	Unknown (14)
Connectivity	Cable Modem (12)	10+ Mbps (91)	DSL (14)	T1 (2)	Unknown (22)
	Slashdot broadcast 12/2002 2pm-10:30pm (total 1316 hosts)				
Region	North America (967)	Europe (185)	Oceania (48)	Asia (8)	Unknown (108)
Background	Home (825)	University (127)	Industry (85)	Government (80)	Unknown (199)
Connectivity	Cable Modem (490)	10+ Mbps (258)	DSL (389)	T1 (46)	Unknown (133)
NAT	NAT (908)	Public (316)	Firewall (92)		

Table 4.2: Host distributions for two broadcast events, excluding waypoints, shown only for a portion of the broadcast.

streams, and information regarding how long users experience periods of bad performance, and the time in between periods of bad performance. Section 4.5.4 presents factors that affect the performance of hosts, and how significant an impact the factors had on overall performance. Section 4.5.5 considers how correlated losses seen by users were with each other, and details regarding catastrophic events that impacted significant portions of the tree. While much of this analysis pertains to application performance, Section 4.5.6 presents metrics that captures the overall quality of the overlay. We summarize our findings in Section 4.5.7.

The data that we use for the analysis is obtained from performance logs collected from hosts participating in the broadcast. Some of the analysis requires logs to be time synchronized. During the broadcast, whenever a host sends a message to the source as part of normal protocol operations (for example, gossip or probe message), the difference in local offsets is calculated and printed as part of the log. In the offline analysis, the global time for an event is reconstructed by adding this offset. We have found that the inaccuracy of not considering clock skew is negligible.

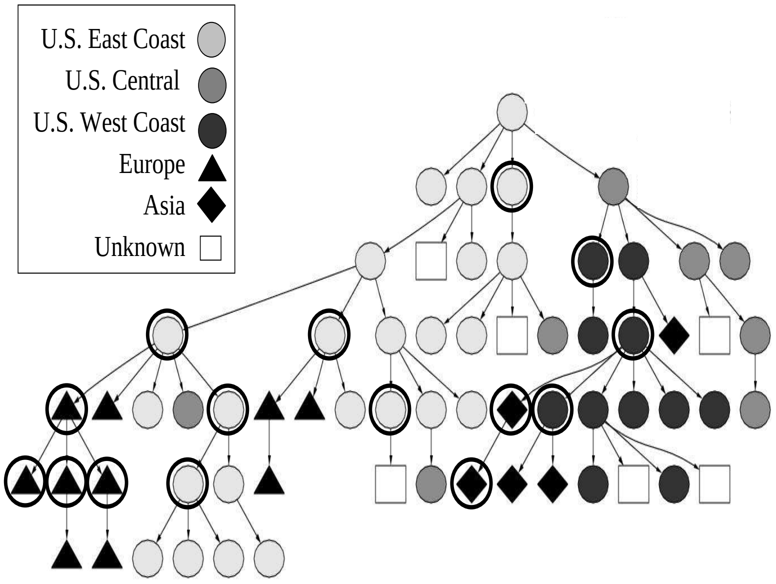

Figure 4.3: Snapshot of the overlay tree during SIGCOMM2002. Participants, marked by geographical regions, were fairly clustered. Waypoints, marked by outer circles, took on many positions throughout the tree.

4.5.1 Average Performance

Figure 4.4 plots the cumulative distribution of mean session bandwidth, normalized to the source rate for the 6 broadcasts. Five of the broadcasts see good performance: more than 90% of hosts get more than 90% of the full source rate in the SIGCOMM2002, Lecture 2, and Lecture 3 broadcasts; more than 80% of hosts get more than 90% of the full source rate in the SIGCOMM2003 and Lecture 1 broadcasts. In the Slashdot broadcast, however, only 60% of the hosts get over 90% of the full source rate.

Table 4.4 summarizes statistics from a feedback form users were encouraged to fill when they left the broadcast. The form required users to rate their satisfaction level for various quality metrics such as ease of setup, overall audio and video quality, frequency of stalls, and duration of stalls. The results are subjective and should be considered in conjunction with the more objective network-level metrics. Further, only 18% of users responded, which further affects the results. Table 4.4 shows that most users were satisfied with the overall performance of the system. Further, more users were satisfied with the overall performance in the SIGCOMM2002 broadcast, which is consistent with the network level metrics in Figure 4.4.

4.5.2 Sensitivity to Stay Time

Our results in Section 4.5.1 presented the performance of receivers that have stayed in the group for at least 1 minute. We consider whether there exists a correlation between how long hosts stay, and the performance they observe. For instance, such a correlation may occur because hosts that see poor performance may leave the group.

Event	Duration (hours)	Incarnations Excluding Waypoints	Mean Session Interarrival Time (sec)	Incarnation Session Duration (minutes)		Entity Session Duration (minutes)		% Eligible Parents	
				Mean	Median	Mean	Median	All hosts	Public hosts
SIGCOMM2002	8	375	83	61	11	161	93	57%	57%
SIGCOMM2003	9	102	334	29	2	71	16	46%	17%
Lecture 1	1	52	75	12	2	26	19	62%	33%
Lecture 2	2	72	120	31	13	50	53	44%	21%
Lecture 3	1	42	145	31	7	42	31	73%	43%
Slashdot	8	2178	17	18	3	11	7	19%	7%

Table 4.3: Summary of group membership dynamics and composition for the 6 larger broadcasts using the system.

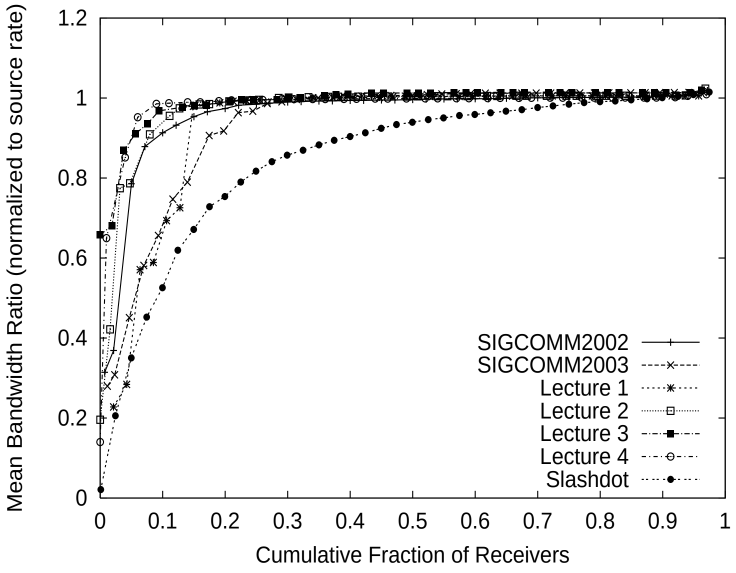

Figure 4.4: Cumulative distribution of mean session bandwidth (normalized to the source rate) for the 6 larger broadcasts.

Figure 4.5 plots the cumulative distribution of mean session bandwidth, normalized to the source rate for the SIGCOMM2002 broadcast. Each curve presents the distribution of performance of hosts, only considering those hosts that have stayed in the group for at least a certain amount of time ($MinStay$). We see that curves corresponding to a $MinStay$ of 15 seconds or more are indistinguishable from each other, however the curve corresponding to a $MinStay$ of 0 has poorer performance. This was primarily because of a set of about 12 hosts that were unable to receive any data, and likely left the group as a result of this.

Figure 4.6 plots a similar set of curves for the Slashdot broadcast. The performance is more sensitive to the $MinStay$ parameter, with the performance of curves corresponding to a $MinStay$ of 2 or more minutes being much better than curves corresponding to a $MinStay$ of 1 minute or less. We investigate this further by considering the convergence time, or the time users take to see good performance. Figure 4.7 plots the performance of only hosts that have a $MinStay$ of at least 5 minutes. All curves correspond to the

	Setup ease	Audio Quality	Video Quality
SIGCOMM2002	95%	92%	81%
Slashdot	96%	71%	66%

Table 4.4: Summary of user feedback for two broadcast events. Each number indicates the percentage of users who are satisfied in the given category.

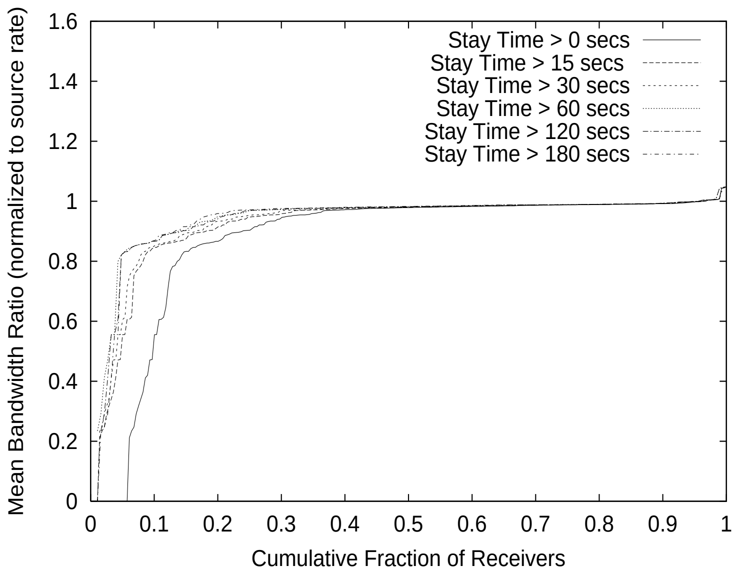

Figure 4.5: Cumulative distribution of mean session bandwidth (normalized to the source rate) for the SIGCOMM2002 broadcast. Each curve corresponds to the hosts that have stayed for at least a certain amount of time.

same set of hosts - but show the cumulative distribution of the mean performance of the hosts over the first X minutes of their sessions for various values of X. We see that the mean performance of the hosts increases with X, and the mean performance over the entire session is significantly better than the mean performance over the first minute.[1]

Our results indicate that in the Slashdot broadcast, there exists a strong correlation between the duration hosts stay and the performance they observe. However, we are unable to precisely decouple cause and effect from our results. On the one hand, we believe a large number of receivers tuned in merely to check the system and were not really interested in the content. These users intrinsically intended to stay for a short time, during which performance had not yet converged, resulting in poor performance. On the other hand, it is also possible that users who otherwise intended to stay longer were impatient and left the system before it had converged to good performance. It is likely that both these factors had played a role - however we are unable to seperate the factors based on the information in our logs.

[1]We believe the larger convergence times in Slashdot are related to the composition of the environment, and discuss this later in the chapter.

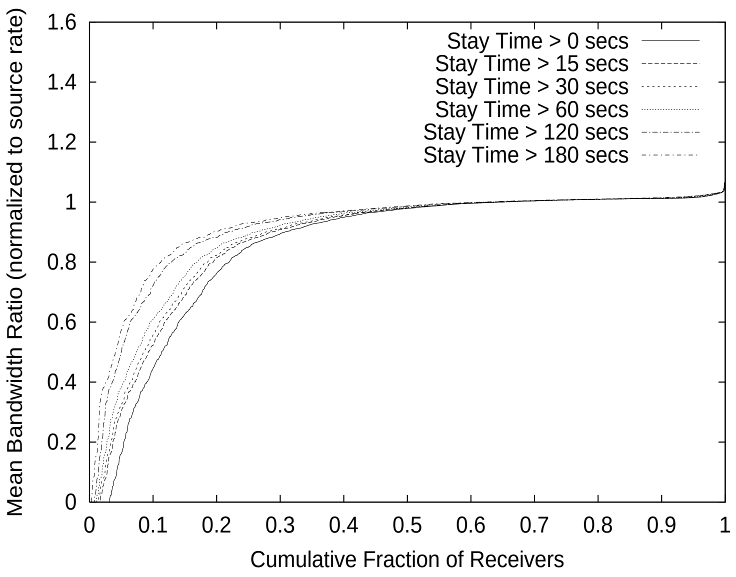

Figure 4.6: Cumulative distribution of mean session bandwidth (normalized to the source rate) for the Slashdot broadcast. Each curve corresponds to the hosts that have stayed for at least a certain amount of time.

4.5.3 Transient Performance of Users

We measure the transient performance based on the application-level losses that users experience. Our system is instrumented with measurement code that logs application loss rate sampled at 5 second intervals. A sample is considered as being a loss if its value is larger than 5% for each media stream, which in our experience is noticeable to human perception. We use three inter-related, but complementary metrics: (i) *session loss* , or the fraction of session for which the incarnation sees loss; (ii) mean interrupt duration; and (iii) interrupt frequency.

Session Loss is computed as follows. If an incarnation participates for 600 seconds, it would have about 120 loss samples. If 12 of those samples are marked as being a loss, then the incarnation sees loss for 10% of its session.

We define an interrupt to be a period of (one or more) consecutive loss samples. *Interrupt duration* is computed as the amount of time that loss samples are consecutively marked as losses. The interrupt durations are then averaged across all interrupts that an incarnation experiences. Note that this metric is sensitive to the sampling period.

Interrupt frequency is computed as the number of distinct interrupts over the incarnation's session duration, and reflects the dynamicity of the environment. A distinct interrupt is determined to be a consecutive period for which the loss samples are marked as a loss. For example, if an incarnation stays for 1 minute, and experiences 2 distinct 5-second interrupts, the interrupt frequency would be once every 30 seconds.

Figure 4.8 depicts the cumulative distribution of the fraction of time all incarnations saw more than 5% packet losses in all three streams in Slashdot and the SIGCOMM2002 broadcast. We only consider incarnations that stay for at least 1 minute to eliminate biases

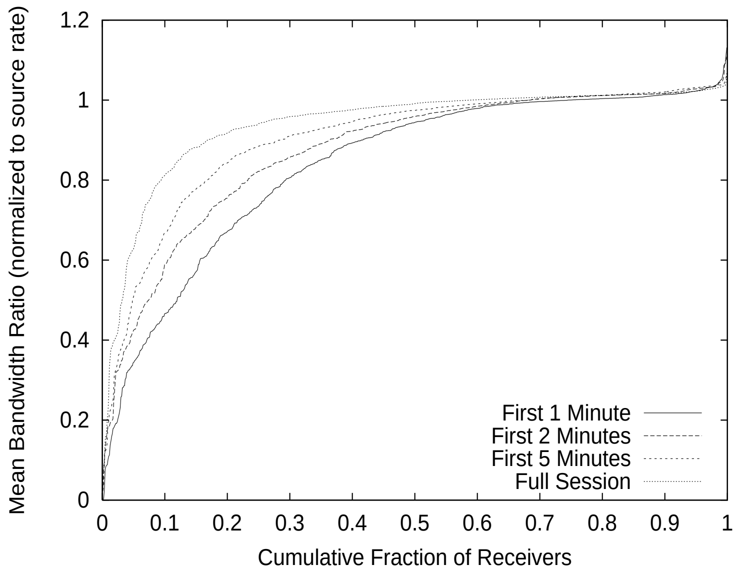

Figure 4.7: Cumulative distribution of mean session bandwidth (normalized to the source rate) for the Slashdot broadcast for hosts that stay for at least 5 minutes. Each curve corresponds to the performance these hosts observe during a certain amount of time from the start of the session.

due to incarnations that have short session durations. For the SIGCOMM2002 broadcast, the performance is good. Over 60% of the hosts see no loss in audio and low quality video, and over 40% of the hosts see no loss in high quality video. Further, over 90% of the hosts see loss for less than 5% of the session in the audio and low quality streams, and over 80% of the hosts see loss for less than 5% of the session in the high quality stream. We will further analyze the performance of the hosts that are seeing the worst performance in Section 4.5.4. For the Slashdot broadcast on the other hand, the low quality video and audio streams see reasonable performance, but the performance of the high quality stream is much less satisfactory. Over 70% of the users see loss for less than 10% of the session in low quality video, but only 50% of users see loss for less than 10% of the session for high quality video. Note that the audio and low quality streams are seeing better performance than the high quality because they are being prioritized in the transmission. For sessions with a high loss rate of high quality video, the low quality one was actually displayed to the user.

Figure 4.9 plots the cumulative distribution of the mean time between successive interrupts seen by each incarnation. In the SIGCOMM2002 broadcast over 80% of hosts see an interrupt less frequent than once in five minutes and 90% see an interrupt less frequent than once in two minutes. In Slashdot, 60% of hosts see an interrupt less frequent than once in five minutes and 80% see an interrupt less frequent than once in two minutes. The higher interrupt frequency with Slashdot probably reflects the more dynamic environment.

Figure 4.10 depicts the cumulative distribution of the duration of interrupts seen by each incarnation. We find that the interrupt duration is almost identical for 5 curves: all 3 streams in SIGCOMM2002, and low quality video and audio in Slashdot. More than 70%

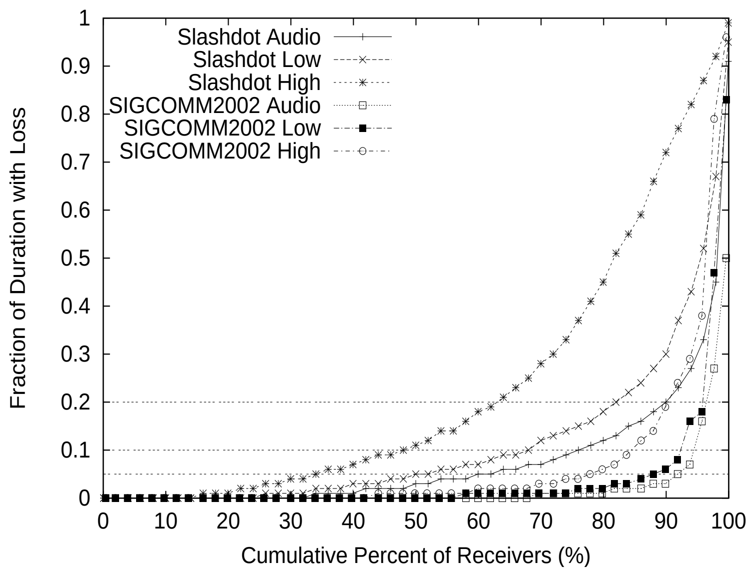

Figure 4.8: Cumulative distribution of fraction of session time with more than 5% packet loss of hosts in the two broadcasts.

of hosts see a mean interrupt duration of less than 10 seconds, and 90% of hosts see a mean interrupt duration of less than 25 seconds for all 5 streams. However, the high quality video in Slashdot sees a pronounced higher interrupt duration, with the last 10% of the hosts seeing a mean interrupt duration of greater than 50 seconds.

4.5.4 Loss Diagnosis

Our results indicate that while the conference-style broadcasts see good performance, the performance with the Slashdot broadcast is less satisfactory particularly with regard to the high quality video streams. Further, even for the conference-style broadcasts, there is a tail, with a small fraction of hosts seeing higher losses and recovery times.

We believe the poorer performance with Slashdot is due to the composition of the environment. With over 66% of the total incarnations behind cable/DSL, and 68% of the incarnations behind NAT/Firewall, a very small fraction of the total group members have the ability at any point in time to take on further children. Further, though the members may be able to support children, their performance may not be good. We consider such resource-constrained regimes in greater detail in Section 4.6, and present design refinements that may help performance in these regimes.

In the rest of this section, we consider the losses that arise in the SIGCOMM2002 broadcast. When evaluating a self-organizing protocol, we need to distinguish between losses that could possibly be fixed by appropriate self-organization techniques from the losses that are fundamental to the system (i.e., those caused by access link capacity limitations, trans-oceanic bottleneck link congestions and local congestions). Further, we are interested in identifying the location of losses in the overlay tree, and attribute causes to the loss. We now summarize steps in our loss diagnosis methodology below:

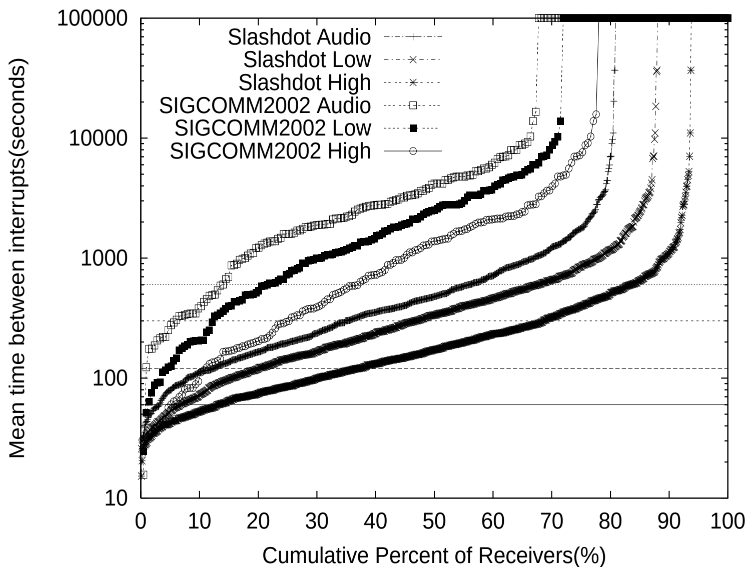

Figure 4.9: Cumulative distribution of time between interrupts

• *Identifying Root-Events:* If a host sees bad performance, then all of its descendants downstream see bad performance. Our first step filters out losses at descendants, and isolates a set of "root-events". If a host sees losses at a particular time, we determine whether its parent saw losses in a 5 second window around that time. This correlation relies on the time synchronization mechanism that we described earlier in the section.
• *Identifying Network Events:* Next, we classify the losses between the host and its parent based on cause. In our system, there are potentially two primary causes: (i) parent leave or death, and (ii) network problems (congestion or poor bandwidth) between the parent and child. There could be other miscellaneous causes such as implementation bugs. Events corresponding to parent leave and death can be identified by examining the parent logs. Implementation bugs are revealed by abnormal patterns we detect during manual verification and analysis of logs. We classify the remaining losses that we are not able to attribute to any known cause as due to network problems.
• *Classifying constrained hosts:* Network losses can occur at several locations: (i) local to the child where a parent change is not needed; or (ii) local to the parent, or on the link between parent and child. As a first step, we identify hosts that see persistent losses near it using the following heuristic. If a host has seen losses for over a significant fraction of the session, if most of the losses are "root-losses" (i.e. the parent of the host does not see loss), and the host has tried several distinct parents during the session, then we decide the host is bandwidth constrained. Inherent here is the assumption that the protocol is doing a reasonable job in parent selection. We note that while this heuristic works well in environments with higher Resource Index, it is not as effective in environments with lower Resource Index. This is because in low Resource Index regimes, it is difficult to distinguish whether a node did not find a good parent because its local resources were limited, or

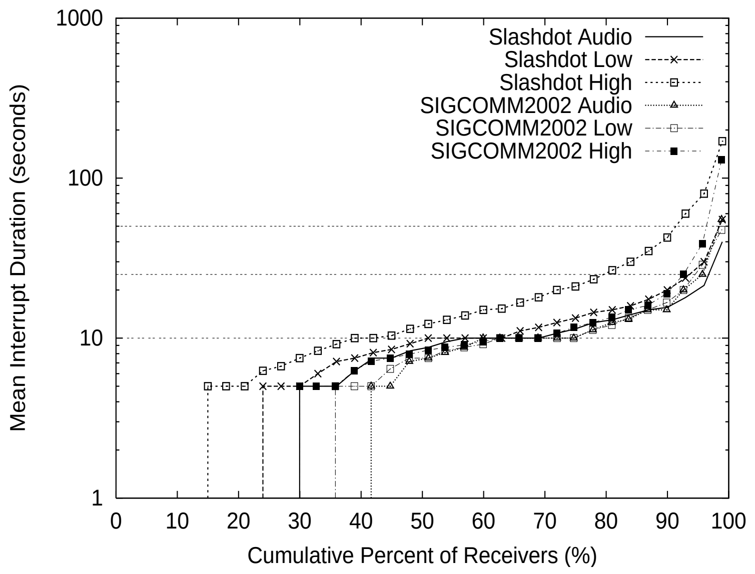

Figure 4.10: Cumulative distribution of mean interrupt duration.

because the choices in the system were fundamentally poor. Finally, we manually verify these hosts and look for other evidence they are constrained (for example, location across a trans-oceanic link, names indicating they are behind wireless links etc.).
• *Classifying congestion losses:* The remaining losses correspond to hosts that usually see good performance but see transient periods of bad performance. If its siblings experience loss at around the same time, it is evidence that the loss is near the parent and not near a child; if a child has made several parent changes during an extended loss period, it is evidence that the loss is near the child. For the events that we are unable to classify, we label them as having "unknown location".

We analyze the data from the SIGCOMM2002 broadcast using the loss diagnosis methodology to get better insight into the tail in performance in Figure 4.8. Figure 4.11 shows the breakdown of all loss samples across all hosts. We find that almost 51% of losses are not fixable by self-organization. 49% corresponded to hosts that were bandwidth constrained, while 2% of losses belonged to hosts that were normally good, but experienced network problems close to them for a prolonged period. 6% of losses corresponded to network events that may be fixable by adaptation, while 18% of losses corresponded to network events that we were not able to classify. Manual cross-verification of the tail revealed about 30 incarnations that were marked as constrained hosts. This corresponded to about 17 distinct entities. Of these, 5 are in Asia, 1 in Europe, 3 behind wireless links, 1 behind a LAN that was known to have congestion issues, and 7 behind DSL links.

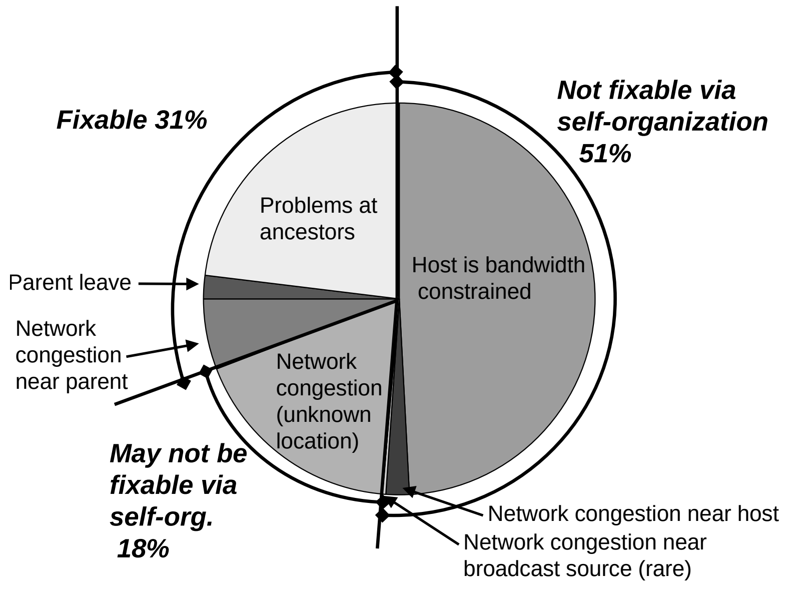

Figure 4.11: Loss diagnosis for SIGCOMM2002.

4.5.5 Correlation of losses among users

Our results so far have focused on the performance of individual users independent of other users. In this section, we consider how losses are correlated across users - that is, how many hosts may simultaneously see loss at a given time instant. This may occur for instance if there is congestion on a link higher up in the overlay tree that affects several descendants.

The session is sampled at periodic intervals, and we estimate the number of members that see loss in each sample. The graph plots the fraction of the total samples for which less than a certain number of hosts see loss, against the number of hosts that see loss. A point (x, y) indicates that there are less than x members that experience loss simultaneously for a fraction y of the total session time. For over 90% of the samples, less than 3 members simultaneously see loss, while for over 99% of the time samples less than 9 members simultaneously see loss. However, there is a tail, which indicates catastrophic events could occur that affect a significant number of members.

Figure 4.13 provides insight on how the catastrophic events are distributed at different times of the broadcast. The X-Axis represents the number of seconds that have elapsed since the broadcast started. The top curve shows the total number of members that are present in the group at any time. The impulses show the number of members that see loss at a given time. For most of the time, there are very few members that see loss, confirming Figure 4.12. There are sharp spikes corresponding to catastrophic events that occur high in the tree, but most of these spikes last a relatively short duration.

Figure 4.13 also indicates that the loss spikes are more concentrated in certain periods - in particular, there is a higher concentration around the time $8800 - 10000$. One possible reason for this is a particularly harsh catastrophic event that occured at around 8800. This event involved an outage on the high bandwidth Internet-2 connection from the source that

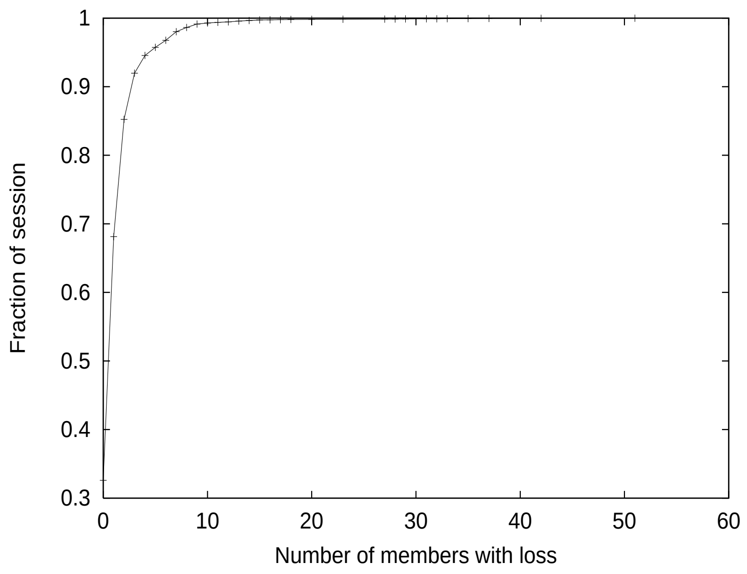

Figure 4.12: The session is sampled at periodic intervals, and we estimate the number of members that see loss in each sample. The graph plots the fraction of the total samples for which less than a certain number of hosts see loss, against the number of hosts that see loss.

lasted nearly 3 minutes. The outage affected 5 of the 6 children of the source, with the unaffected child being in Europe. The protocol reacted by having all these children pick parents from nodes present in the subtree rooted at the European node. After a while the outage cleared and nodes in the US moved back to the source. Figure 4.15 provides confirmation of this episode by plotting the average delay of the recepients from the source as a function of time. The delay here refers to the sum of the delays of all the overlay links from the source to the receiver, measured from protocol operations. There is a sharp spike around time 8800 - this corresponds to a tree with the source having a single child presented in Europe. While the actions surrounding the event happened over $100 - 200$ seconds, a subsequent period of over 1000 seconds appears to have a higher frequency of high loss instances. It is possible the earlier actions impacted the tree structure in a way to make it more susceptible to failures over the longer term. However, we are unable to find direct evidence of this and distinguish this from pure random chance.

4.5.6 Resource Usage

Our results so far focus on application-level losses. We now consider how efficiently the constructed overlay uses the network resources in the system. Consider Figure 4.3 again, which plots a snapshot of the overlay during the SIGCOMM2002 broadcast. The shape and color of each node represents the geographical location of the host as indicated by the legend. We see that the protocol does a reasonable job of clustering nodes that are in the same geographical location.

To more formally evaluate the protocol, we consider the resource usage metric defined

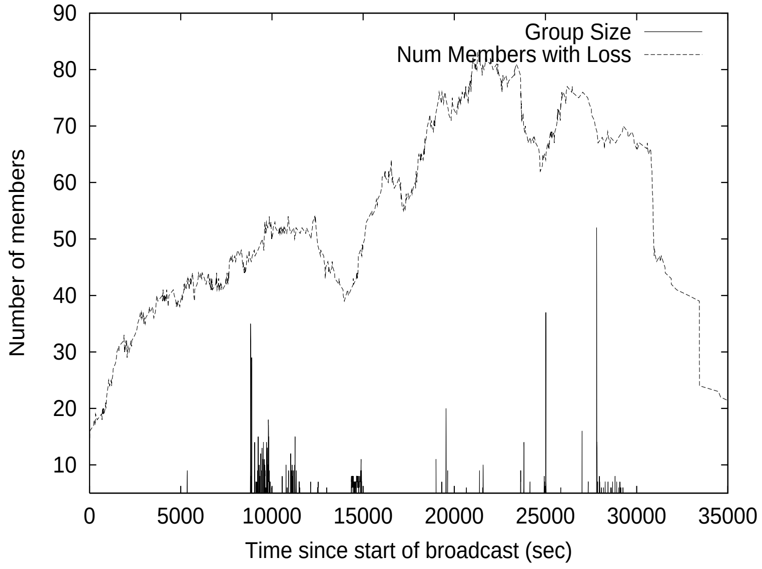

Figure 4.13: The spikes show the number of members that see loss at a particular time instant in the broadcast as a function of the time that has elapsed (in seconds) since the beginning of the broadcast. The total number of members in the group at that instant is also plotted.

in Chapter 2. We compute the resource usage of an overlay tree to be the sum of the delays of each of the overlay links. We obtain information about delays between members from the probe messages they use which are part of the protocol. Further, we consider the ratio of this value to the resource usage of the minimum spanning tree constructed on the same set of nodes. Figure 4.14 plots the ratio at different times during the broadcast. For most of the broadcast, the resource usage is small, and varies between 2 and 3. Note that the minimum spanning tree though optimal in resource usage may not necessarily be a feasible data delivery tree, as it may not honor the outgoing bandwidth constraints of nodes. Finally, it is possible that at much larger scales, the protocol is less efficient at clustering members given the random nature of our group management protocol. This can be potentially be improved by having the source provide nodes information about geographically closer nodes, through mechanisms like network coordinates of nodes [71].

4.5.7 Summary and Discussion of Results

In this section, we presented results analysing the performance of the ESM system from several perspectives in many real broadcasts. We summarize our key results:

• For the conference/lecture style broadcasts, the system has provided consistently good application level performance. For all these broadcasts, the system provided an average throughput of more than 80%−90% of the full source rate. In the SIGCOMM2002 broadcast, over 90% of the hosts see loss for less than 5% of the session in the audio and low quality

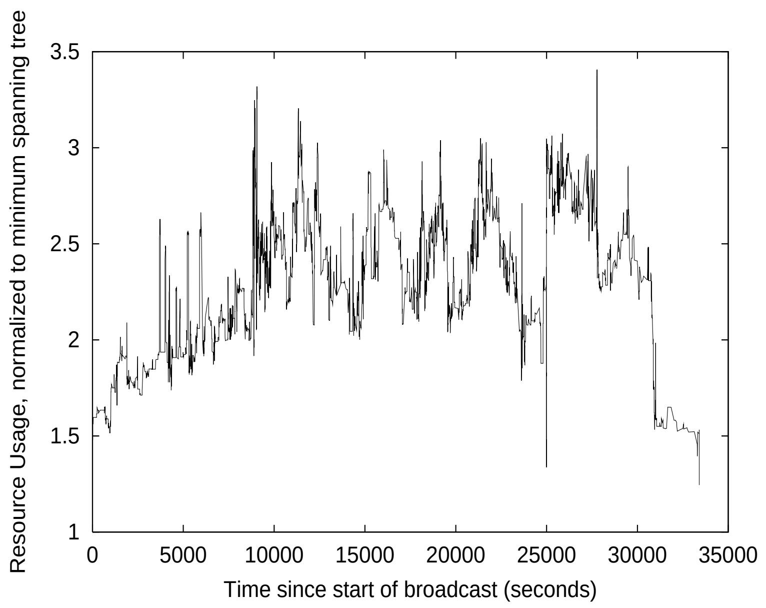

Figure 4.14: Resource Usage as a function of time since start of broadcast (in seconds)

streams, and over 80% of the hosts see loss for less than 5% of the session in the high quality stream.
• We presented a loss diagnosis procedure to determine key causes for losses. In the SIGCOMM2002 broadcast, over 51% of the losses are due to constraints close to the host - hosts that are not able to receive the full throughput during the session, or hosts that normally see good performance, but may see periods of bad performance close to them.
• With the Slashdot broadcast, users see good performance with audio and low quality video, but the performance of high quality video is not as satisfactory. While over 70% of the users see loss for less than 10% of the session in low quality video, only 50% of users see loss for less than 10% of the session for high quality video. The key reason for this is the constitution of the environment - a much higher fraction of hosts behind cable modem, DSL and NAT results in a much smaller fraction of nodes being eligible parents.
• We investigated how correlated losses among users are. While most of the time there are very few users that see losses, there are occasional periods due to catastrophic events where a large number of users see loss. In the SIGCOMM2002 broadcast, we saw the catastrophic events being more concentrated at particular points in time. While it is possible mechanisms in the protocol that result in better quality tree structures may have helped alleviate this, we are unable to find strong evidence to support this.
• We measured the *Resource Usage* of trees constructed by the protocol, and find this to range between a factor of 2 and 3 times the resource usage of the minimum spanning tree. We note that the minimum spanning tree itself may not represent a feasible data delivery tree. Finally, it may be possible to further reduce the resource usage using explicit clustering mechanisms in the protocol.

Our system represents a particular point in the design space of solutions for End System

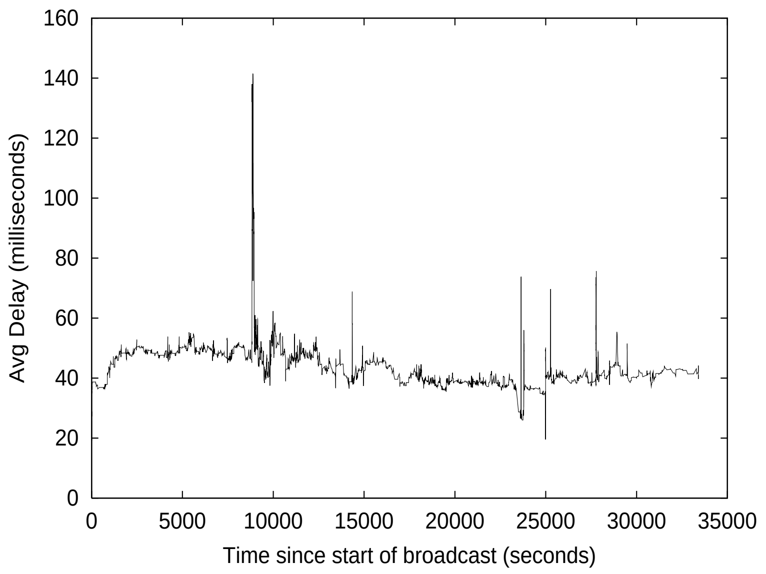

Figure 4.15: Average delay experienced by receivers as a function of time since start of broadcast (in seconds). Note that the delay merely refers to the sum of the UDP delays of each overlay link, and does not correspond to actual delay seen by a data packet (given the use of TCP on each link).

Multicast. Further, as discussed earlier, our overall approach while designing the system has been to emphasize putting together a complete prototype, while adopting simple or natural design choices for any given component. Our results demonstrate that fairly simple designs with ESM go a long distance towards enabling an interesting class of broadcasting/conferencing applications on the Internet. While our results show the promise of ESM, there are several directions in which the design of the system can be improved in the future. These include the construction of more resilient data delivery structures rather than trees, incorporating error-recovery mechanisms to exploit buffers at end systems, refining the group management algorithms to achieve better clustering, and addressing concerns regarding flash crowds. While all these are important directions, the key challenge that our deployment experience has highlighted is the need to provide good performance in *resource-constrained* environments like Slashdot, where a large fraction of hosts are behind NATs, and bandwidth-constrained access technologies like cable modem and DSL.

4.6 Design Lessons

In this section, we present design refinements that can enhance performance in environments like Slashdot that are resource-constrained. To more formally captures the resource availability in the environment, we introduce a metric that we call the *Resource Index.*

We define the *Resource Index* to be the ratio of the total outgoing bandwidth of the system to the total incoming bandwidth in the system. A *Resource Index* of 1 indicates that the system is saturated, and a ratio less than 1 indicates that not all the participating

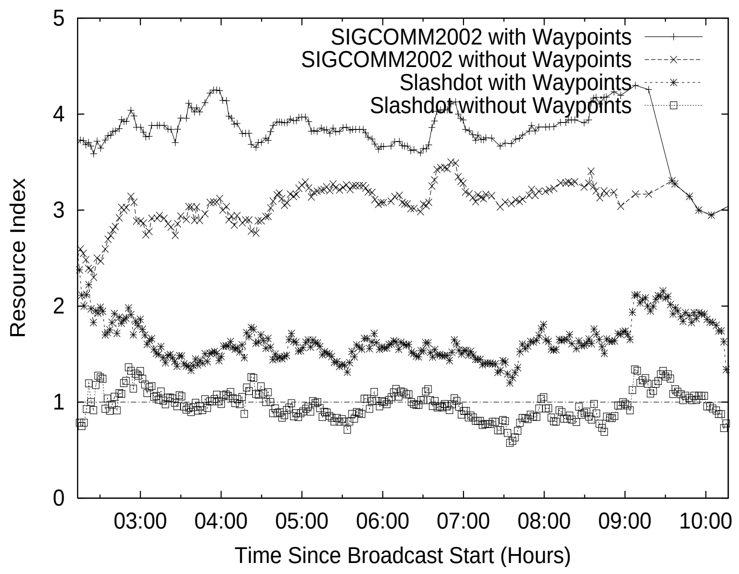

Figure 4.16: Resource Index as a function of time with and without waypoint support for the two broadcasts.

hosts in the broadcast can receive the full source rate. As the *Resource Index* gets higher, the environment becomes less constrained and it becomes more feasible to construct a good overlay tree. The definition of *Resource Index* has been extended to incorporate the connectivity constraints of NATs and firewalls, by only considering free slots available for NAT hosts. See [12] for details. The *Resource Index* only captures the availability of resources in the environment, and does not account for factors such as performance of Internet paths. Further, the index is computed assuming global knowledge, but in practice, a distributed protocol may not be able to use the resources as optimally as it could have.

4.6.1 On-demand Waypoint Invocation

Our broadcasts employed Planetlab hosts that we call waypoints in addition to the actual user machines, as discussed in Section 4.4. We study the role played by the waypoints in our broadcasts.

Figure 4.16 plots the *Resource Index* in the Conference and Slashdot broadcasts as a function of time of the broadcast. There are two curves corresponding to each broadcast - one corresponding to the actual set of hosts present in the broadcast, and one without the waypoint machines we used. With the waypoints, the *Conference* broadcast has a Resource Index of 4, potentially enough to support 4 times the number of members. Further, the *Conference* broadcast had enough capacity to sustain all hosts even without waypoint support. In fact, all our broadcasts besides *Slashdot* had sufficient resources among participating hosts and did not require the support of waypoints. In one of the lecture broadcasts, all the waypoint left simultaneously in the middle of the broadcast due to a configuration problem, and we found that the system was able to operate well without the waypoints.

In contrast, the *Slashdot* broadcast has a Resource Index that is barely over 1. Further,

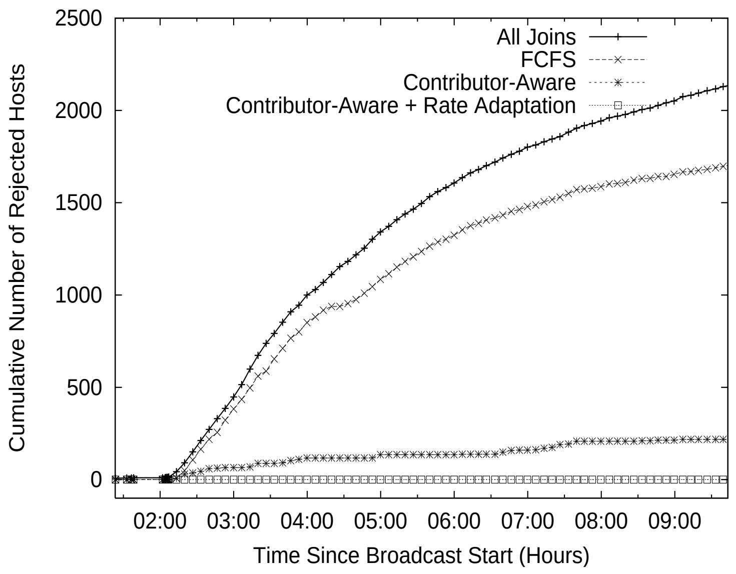

Figure 4.17: Number of rejected hosts under three different protocol scenarios in the simulated Slashdot environment.

the *Resource Index* was much lower without waypoint support, and occasionally dipped below 1. This indicates that it was not always feasible to construct an overlay among all participating hosts which could sustain the source rate. Dealing with such environments can take on two complementary approaches: (i) designing techniques that can enable good performance in purely application end-point architecture, even in the absence of waypoints (which forms the thrust of the subsequent sections); and (ii) use a waypoint architecture, with the insight that waypoints may not be needed for the entire duration of the broadcast, and can be invoked on-demand. In our future deployments, our objective is to explore both approaches and gradually decrease the dependence on waypoints, using them as a back-up mechanism only when needed.

In the long-term, waypoint architectures may constitute an interesting research area in their own right, being intermediate forms between pure application end-point architectures which rely only on end-user resources, and statically provisioned infrastructure-centric solutions. The key aspect that distinguishes waypoints from statically provisioned nodes is that the system does not depend on these hosts, but leverages them to improve performance.

4.6.2 Contributor-Aware Overlays

If the Resource Index dips below 1 like in the Slashdot broadcast without waypoints, the system must reject some hosts or degrade application quality. In this section, we evaluate performance in terms of the fraction of hosts that are rejected, or see lower application quality. We consider three policies. In the *First-Come-First-Served (FCFS)* policy that is currently used in our system, any host that is looking for a new parent, but finds no unsaturated parent is rejected. In the *Contributor-Aware* policy, the system distinguishes between two categories of hosts: contributors (hosts that can support children), and free-riders (hosts that cannot support children). A contributor C that is looking for a new parent may preempt a free-rider (say F). C can either accommodate F as a child, or kick it out of the system if C is itself saturated. This policy is motivated by the observation that preferentially retaining contributors over free-riders can help increase overall system resources. Finally, we consider *Rate-Adaptation* where a parent reduces the video rate to existing free-riders in order to accommodate more free-riders. For example, a parent can stop sending the high quality video (300 kbps) to one child, and in return, support three additional 100 kbps children. This policy is an example that not only differentially treats hosts based on their capabilities, but also exploits application knowledge.

We evaluate the potential of these policies by conducting a trace-based simulation using the group membership dynamics pattern from the Slashdot broadcast. We retain the same constitution of contributors and free-riders, but remove the waypoints from the group. We simulate a single-tree protocol where each receiver greedily selects an unsaturated parent, and we assume global knowledge in parent selection. If there is no unsaturated parent in the system, then we take action corresponding to the policies described above. Figure 4.17 shows the performance of the policies. We see that throughout the event, 78% of hosts are rejected using the *FCFS* policy. *Contributor-Aware* policy can drastically reduce the number of rejections to 11%. However, some free-riders are rejected because there are times when the system is saturated. With the *Rate Adaptation* policy however, no free-rider is rejected. Instead, 28% of the hosts get degraded video for some portion of the session.

Our results demonstrate the importance and potential of contributor-aware rejection and rate adaptation. A practical design has to deal with many issues, for example, robust ways of automatically identifying contributors, techniques to discover the saturation level of the system in a distributed fashion, and the trade-offs in terms of larger number of structure changes that preemption could incur. We are currently in the process of incorporating these policies in our design and evaluating their actual performance.

4.6.3 NAT-Aware Overlay Construction

Hosts behind NATs and firewalls can constitute an overwhelming fraction of the total hosts in the Internet (for example, 50%-70% of the hosts in *Slashdot*). This significantly lowers the *Resource Index* given that hosts behind NATs cannot choose other hosts behind NATs as parents.

Two modifications to our system can help improve performance with NATs. A first modification involves using UDP as the transport protocol for data delivery rather than TCP. This is because a UDP-based transport protocol can enable connectivity between hosts behind NATs if one of them is a full-cone NAT. *Full Cone NATs* can receive incoming packets to a port from any arbitrary host once it sends a packet on that port to any destination. Such hosts are in contrast to *Symmetric NATs* which allow incoming packets only from the host that it has previously sent a packet to. A second modification involves making the self-organizing protocol explicitly aware of NAT/firewalls. In particular, if public hosts preferentially choose NATs as parents, then more resources at the public hosts

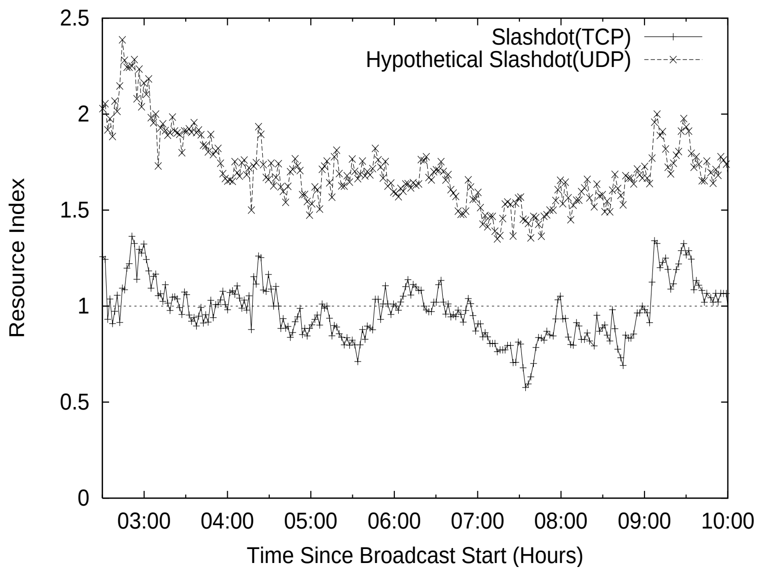

Figure 4.18: Resource Index comparison of two connectivity solutions for NAT/firewall: (i) Slashdot (TCP), (ii) Hypothetical Slashdot (UDP).

themselves are available for NATs/firewalls.

We evaluate the potential of these two design improvements in the *Slashdot* broadcast. Figure 4.18 shows the Resource Index for the system for the various design alternatives as a function of time, again omitting waypoint hosts. The lowest curve corresponds to Resource Index with NAT-Aware protocol structuring, and a TCP-based protocol. The upper curve corresponds to the Resource Index with a UDP-based protocol and a NAT-aware protocol construction. The upper curve represents an improvement of 74%, indicating that the combination of the two techniques has the potential to significantly improve the *Resource Index*.

4.7 Summary

This chapter presents early experience with an operational broadcasting system based on End System Multicast. To our knowledge this is among the first reports on experience with real application deployment based on overlay multicast, involving real users.

Our emphasis while designing the system has been to build a completely operational prototype, tackling a wide range of issues in the process. Many of the system components present interesting design decisions – our overall approach has been to adopt simple design choices, with a view to revisiting the decisions in the light of future experience.

This thesis has led the design and implementation of Sparta - the self-organizing protocol in the system. The design of Sparta is *integrated* with the overall objectives of validating the ESM architecture, and building an operational system. Sparta builds on our prior experience with Narada. However, it employs different group management algorithms as compared to Narada given the shift in focus from smaller-scale multi-source conferencing applications to larger-scale single-source broadcasting applications. Further, implementing the protocol in the context of a real system has led to several refinements of the ideas in Narada, and has led to a unique implementation-oriented perspective on protocol design.

Our experience with the system has included several positives, and has taught us important lessons which we summarize below:

• *ESM is easy to deploy:* In less than a year, we have broadcast tens of events and our system has been used by thousands of users. The system is satisfying the needs of real content publishers and viewers. The users have been from diverse geographical locations, and from home, commercial and educational environments. We believe the deployment achieved surpasses deployment using the IP-Multicast based MBone experimental network, both in terms of number of participants, and with respect to diversity of participants. The Mbone, even in its prime, was primarily restricted to university users.
• *ESM provides good application performance:* We have provided a detailed characterization of the performance of the system with a range of metrics. Our experience with several conference/lecture-type broadcasts indicate that our system provides good performance to users. In such environments, we consistently observe that over $80 - 90\%$ of the hosts see loss for less than 5% of their sessions. Further, hosts that perform poorly are typically bandwidth constrained hosts. Even in a more extreme environment like *Slashdot* , users see good performance in audio and low quality video. These results indicate that reasonably simple designs adopted in the system can go a long way towards enabling good performance with ESM.
• *Environments with low Resource Index are challenging:* A key issue with our design that has been highlighted through experience is the performance in regimes with low Resource Index. We considered a range of refinements that can help address the issue based on exploiting heterogeneity in node capabilities through differential treatment, NAT-aware overlay construction, and on-demand waypoint invocation. We hope to implement these refinements and gain more experience in such regimes in the future.

Overall, our experience demonstrates the potential of End System Multicast as a cost-effective alternative for enabling Internet broadcast, and significantly increases the credibility of the architecture.

Chapter 5

The Impact of Heterogeneous Bandwidth Constraints on Protocol Design

Our experience with a broadcasting system (Chapter 4) highlighted the importance of leveraging the heterogeneity in node bandwidth constraints while deploying ESM in environments where the overall bandwidth resources in the system are scarce. In this chapter, we systematically study the implications of heterogeneous bandwidth constraints of nodes on multiple existing protocol designs for End System Multicast. Our work is set in the context of bandwidth-intensive and non-interactive applications in environments with little dedicated infrastructure support.

Figure 5.1 summarizes the constitution of hosts in studies with operationally deployed overlay systems [12, 58] including our own, and highlights the significant heterogeneity present in Internet environments today due to presence of various access technologies like cable modem, DSL and Ethernet. However, the implications of such heterogeneity on protocol designs for overlay multicast has received limited attention in the community, where the emphasis has primarily been on issues such as scalability and on delay-based metrics. Our focus in this chapter is on the *outgoing bandwidth* of nodes. This is distinguished from heterogeneity in the incoming, or receiving bandwidth, which has been widely studied [64, 42]. In the rest of the chapter, we use the term "bandwidth" to refer to the "outgoing bandwidth" unless otherwise mentioned.

Two classes of designs have emerged for overlay multicast: (i) *Performance-centric*; and (ii) *DHT-based*. Performance-centric protocols are directly optimized for the application at hand, and the primary consideration while adding links in the overlay topology is application performance. In contrast, DHT-based protocols focus on maintaining a structure based on a virtual id space. The objective is to support multiple applications simultaneously, and amortize costs across the applications [55, 37]. We analyze representative examples of protocols from each class to understand the impact of heterogeneous bandwidth constraints of nodes. We note that our nomenclature merely refers to the primary criterion in selection of links in the overlay topology, and does *not* imply worse performance for DHT-based approaches.

We demonstrate through simulations that in typical deployment settings where the bandwidth resources in the system are scarce, it is critical to treat nodes differently based on their bandwidth constraints. We show that it is feasible to extend performance-centric protocols to incorporate such differential treatment. To this end, we incorporate appropriate techniques in HMTP [76], a representative performance-centric protocol, and show their potential using a systematic evaluation. To the best of our knowledge, this is the first systematic study of techniques for bandwidth-based differention in overlay multicast protocols.

We also consider Scribe [37], a representative and relatively mature DHT-based protocol for overlay multicast. Our analysis reveals that imposing bandwidth constraints on Scribe can result in the creation of a significant number of *non-DHT links,* that is, links that are present in the overlay tree but are not part of the underlying DHT. Non-DHT links are undesirable because they restrict the benefits of the route convergence and loop-free properties of DHT routing, and incur maintenance costs in addition to that of the DHT infrastructure. We find that a key cause for the creation of non-DHT links is the mismatch between the id space that underlies the DHT structure and node bandwidth constraints. We discuss potential ways of addressing the issues.

The rest of the chapter is organized as follows. Section 5.1 provides background and related work. Section 5.2 presents our evaluation framework. Section 5.3 presents issues with heterogeneous bandwidth constraints, and how they may be addressed in performance-centric protocols. Section 5.4 discusses the same issues with DHT-based protocols.

5.1 Background and Related Work

5.1.1 Two classes of protocol designs

We broadly classify protocols for overlay multicast as falling under one of two categories: (i) Performance-centric protocols; and (ii) DHT-based protocols.

In performance-centric protocols, the focus is on constructing overlay topologies where neighbor relationships are primarily governed by performance. Such protocols typically incorporate special group management mechanisms to enable each node to maintain knowledge of a subset of group members. When a parent dies, or bad performance is observed, nodes switch to a parent they believe can provide better performance. Nodes may also perform periodic probes for locating better parents. Both protocols presented in this thesis - the Narada protocol in Chapters 2 and 3, as well as the broadcasting protocol in Chapter 4 - are examples of performance-centric protocols. Other examples of such protocols include Scattercast [10], Yoid [24], NICE [57], Overcast [34], and HMTP [76].

In contrast, in DHT-based protocols, such as Delaunay Triangulations [36], CAN-multicast [49], Bayuex [66] and Scribe [37], members are assigned addresses from an abstract coordinate space such as a ring, torus or hypercube. By creating neighbor links based on these addresses, a structured overlay is created which enables scalable and efficient unicast routing based on the node identifiers. These unicast routes are then used for creating multicast distribution trees.

Two principal reasons have been advocated for a DHT-based approach. First, such an approach provides a generic primitive that can benefit a wide range of applications besides overlay multicast, such as file-sharing. Second, the same DHT-based overlay can be used to simultaneously support and maintain a large number of overlay applications and multicast trees. This could help keep the costs low as compared to constructing and maintaining many individual overlays.

While DHT-based approaches have several advantages, a key concern is application performance, given that selection of overlay links is not dictated by performance alone. Studying the performance potential of DHT-based overlays, and mechanisms to enable good performance with DHTs is an active and ongoing area of research. Much of the work to date has focused on delay-based metrics. More recent work has investigated the impact of heterogeneity in nodes for file-sharing applications [19]. In contrast, this chapter focuses on concerns specific to heterogeneous bandwidth constraints of nodes unique to overlay multicast.

5.1.2 Handling of heterogeneous bandwidth constraints

Issues related to the heterogeneity in the bandwidth constraints of nodes have not been considered by many of the most frequently cited and used self-organizing protocols both in the DHT and non-DHT space. This includes protocols such as Yoid [24], Narada [11], Overcast[34], Scattercast[10], ALMI [45], Scribe [37], CAN [49], Bayeux [66], Delaunay Triangulations [36], NICE [57], HMTP [73], SplitStream [8] Coopnet [43] and Bullet [14]. The focus of most of this work has been on algorithms for scalable group management, ensuring low delay penalties, and redundancy of data delivery.

Dealing with heterogeneity in the *receiving* capabilities of nodes through mechanisms like layered coding has been widely studied [64, 42]. In contrast, our focus is on heterogeneity in the *outgoing bandwidth* of nodes.

The key idea underlying the techniques considered in this chapter is *preemption,* or the ability of a node to displace another node in the overlay tree. This notion has been employed by other works [8, 40, 37]. However, the preemption heuristics used in [8, 37] are based on delays or prefix matches of nodes, in contrast to this chapter where the emphasis is on preemption based on bandwidth constraints. Second, none of these works consider the impact of preemption on tree depth, nor systematically study the trade-offs associated with the disruptive nature of preemption. In contrast, this chapter highlights the critical need of bandwidth-based preemption, and investigates issues related to incorporating these techniques in different classes of protocols.

5.2 Evaluation Framework

The setting we consider is that of unidirectional multicast from a single source to a set of receivers. Such unidirectional flow of data would typically correspond to non-interactive applications such as video broadcasting which do not place a tight constraint on the end-to-end latency. We assume a constant bit rate (CBR) source stream, which is a good approximation for streaming video sources in the Internet. We assume an environment where only nodes that are interested in the content at any point in time are members of the distribution tree and contribute bandwidth to the system. We assume a single distribution

Event	Low Speed 100Kbps (deg. 0)	Medium Speed 1.5Mbps (deg. 2)	High Speed 10Mbps (deg. 10)	Avg Deg
SIGCOMM2002 [12]	22%	2%	76%	7.64
Slashdot [12]	74%	4%	22%	2.28
Gnutella [58]	65%	27%	8%	1.34

Figure 5.1: Constitution of hosts from various sources. SIGCOMM2002 and Slashdot refer to two different broadcasts with our operationally deployed ESM Broadcasting system (Chapter 4). Gnutella refers to a measurement study of peer characteristics of the Gnutella system.

tree spanning the set of interested nodes.

5.2.1 Models

We describe key characteristics of the environment that we model and that we believe are important to consider while designing protocols for overlay multicast:

• *Per-node bandwidth constraints:* We use the outgoing bandwidth limit of each host to determine its *degree* or *fanout* in the overlay multicast tree, i.e., the maximum number of children that it can forward the stream to. We make the simplifying assumption that the outgoing bandwidth limit of a host is a fixed quantity that is a property of the host itself regardless of the peer that it is communicating with. This models the common case where the bottleneck for all communication is the access link to the wide area network.
• *Heterogeneity in node constraints:* Our model of heterogeneity in node degrees is derived from realistic application bandwidth requirements and measurements of host bandwidth constraints. Typical streaming video rates on the Internet today are of the order of several hundred kilobits per second [12]. On the other hand, using data from measurement studies [58] and real Internet broadcast events [12] summarized in Figure 5.1, the host connectivity can be categorized into: (a) constrained links such as cable and DSL (few hundred Kbps), (b) intermediate speed links such as T1 lines (1.5 *Mbps*), and (c) high-speed links (10 *Mbps* or better). Based on these observations, we quantized the degrees of the low, medium, and high speed hosts to 0, 2, and 10. The degree 0 nodes are termed *non-contributors.* We note that for higher speed connections, the degree is likely to be bounded by some policy (in view of the shared nature of the links) rather than the actual outgoing bandwidth. While host access link bandwidths are likely to change over time, we believe that there will still be a significant range of bandwidths and that application data rates will still be significant relative to host bandwidths.
• *Bandwidth resources in the environment:* We define the *Average Degree* of the system to be the the total degree of all nodes (including the source) divided by the number of receivers (all nodes but the source). An average degree less than 1 indicates that the system is fundamentally constrained and that some receivers would have to be rejected. On the other hand, an average degree greater than or equal to 1 indicates that it is feasible to

build a tree. The larger the index, the greater is the "excess capacity" of the system. In this chapter, we focus on environments with an average degree greater than 1. One could deal with more constrained environments by employing a multi-rate model where the source rate is lowered, sacrificing quality, to accommodate all receivers, but we do not consider such adaptation in this chapter.

• *Group Dynamics:* We use a Poisson arrival pattern and a Pareto-distributed stay time for clients. These choices have been motivated by group dynamics characteristics observed from our deployment experience described in Chapter 4, and Mbone measurement traces [5]. Unless otherwise mentioned, our experiments last for a duration of 1000 seconds, and assume a mean arrival rate of 10 joins per second. Further, our experiments assume a mean stay time of 300 seconds, a minimum stay time of 90 seconds, and a parameter of $\alpha = 1$ in the Pareto distribution. We note that this corresponds to a steady state group size of about 3000 members.

• *Network Model:* Given that our focus is on bandwidth-sensitive and non-interactive applications, we consider a uniform-delay network model throughout this chapter.

5.2.2 Metrics

Our primary metric for evaluating protocols in this chapter is the *depth* of the multicast distribution trees. We believe this metric significantly influences application performance. As discussed in Chapter 4, the performance seen by a node in an overlay multicast application depends on two factors: (i) the frequency of interruptions due to the failure of an ancestor, or due to congestion on an upstream link; and (ii) the time it takes a protocol to recover from the interruptions. The frequency of interruptions a node experiences in turn depends on the number of ancestors the node has, or the depth of the node in the tree.

In addition to the depth metric, we have considered several other metrics in this chapter. Section 5.3.2 considers the *loss rate* metric,[1] and the impact of reducing depth on loss. We consider a metric called *rejection rate* in Section 5.3.2, and a metric related to the *fraction of non-DHT links* in DHT-based solutions in Section 5.4. We will define these metrics in the appropriate sections.

5.3 Performance-Centric Protocols

In this section, we discuss the issues raised by heterogeneous node constraints to protocols for overlay multicast. We also consider and evaluate the feasibility of extending performance-centric protocols with appropriate techniques to address the issues. We study the same issues in the context of DHT-based approaches in Section 5.4.

While much of the insight in this section applies to any performance-centric protocol, we demonstrate our ideas using HMTP [76]. HMTP is a performance-centric protocol that honors per-node bandwidth constraints, but does not treat nodes differently based on their bandwidth constraints. We further justify our choice in Section 5.3.1.

[1]When interruptions are only due to node departures, we can show $\text{Loss Rate} \approx \frac{\text{depth}}{\text{stay time}} *$ reconnection time, where *stay time* is the mean duration of a node's participation in the multicast tree.

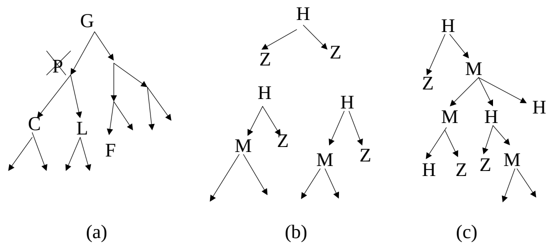

Figure 5.2: Need for differential treatment. (a) Maintaining balanced trees is not easy even with homogeneous degree constraints. If P dies, the depth of L and descendants is affected (Sec 5.3.3). (b) Entire subtrees could be rejected when the subtree connected to the source is saturated with non-contributors. H,M, and Z represent nodes of high degree, medium degree and non-contributors respectively; (c) Depth can be poor with heterogeneous degree constraints

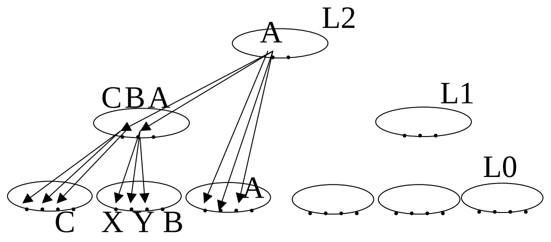

Figure 5.3: An example structure constructed by NICE

5.3.1 Honoring per-node constraints

Our choice of HMTP was motivated by two reasons: (i) it is designed for large-scale groups and has scalable algorithms for group management unlike Narada (Chapter 2); and (ii) it has clear well-defined heuristics to honor per-node degree constraints of nodes. We describe NICE [57] and Overcast [34], two influential protocols that do not honor per-node constraints, and then describe the mechanisms that HMTP [76] adopts to deal with the issues.

To understand the impact of degree constraints on overlay tree construction, consider Figure 5.2(a) that shows a snapshot of an overlay tree. Assume that each node has a degree constraint of 2. If node P dies, one of its children (say C) can take its spot and choose G as its parent. However, node L can neither choose G, nor choose C as its parent. Similarly, when a node joins the group, it might contact the source, but find the source saturated. Thus, any protocol designed with the requirements of per-node constraints must have appropriate mechanisms for dealing with these scenarios.

The NICE protocol, targeted for low latency (rather than high bandwidth) data streaming applications constraints, organizes receivers into a set of layers. Hosts in each layer are partitioned into a set of clusters, each of which has a size between k and $3k-1$. Due to the fact that: (i) a host is a cluster-head at every layer below its top-most cluster; and (ii) a head of a cluster must forward traffic to every member of the cluster, the NICE protocol places an uneven burden on its nodes. For example, the root of the tree forwards data

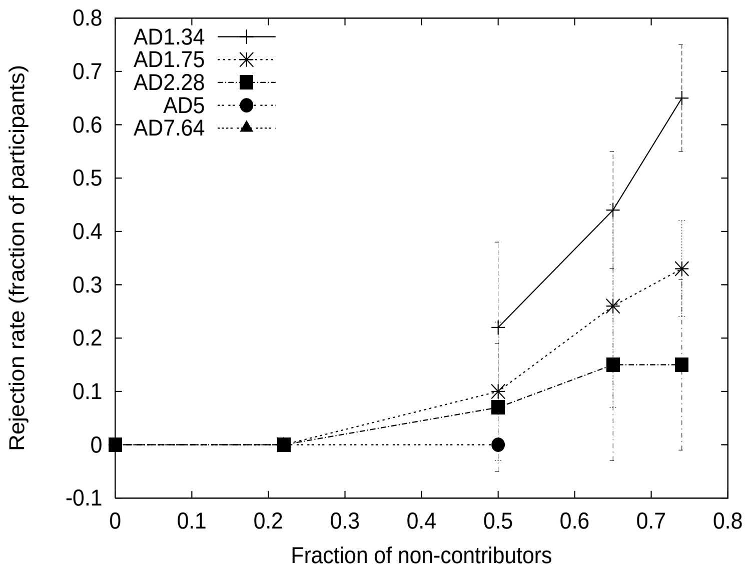

Figure 5.4: Rejection-rate Vs. fraction of non-contributors for various average degree values

to $O(logN) * k$ participants. An example structure that NICE may construct is shown in Figure 5.3. To see how the lack of per-node degree constraints simplifies the design of NICE, consider a situation where node A dies. This results in the promotion of one of its siblings (say B) to the next higher level. B supports its existing children (X, Y), as well as other children of A (say C), because the promotion of B to a higher level enables it to accommodate additional children.

The Overcast protocol [34] designed for infrastructure-centric (as opposed to application end-point) environments, does not enforce any hard degree constraint. For example, in Figure 5.2(a), the failure of node P results in both C and L choosing grand-parent G. When a parent is overloaded, a "lazy migration" procedure is adopted where nodes become children of their old siblings. Such an approach can involve transient overloads that can be of concern in application end-point environments.

We now describe the mechanisms HMTP [76] adopts to handle per-node constraints. A node that joins the source conducts a depth-first style search beginning from the root until it can locate a node with free capacity to attach to. When a node is disconnected from the overlay, for example due to the death of its parent, it contacts the grandparent and conducts a similar depth-first search. For example, in the example above, node L begins a depth first search operation at G, picking F as parent. HMTP also contains heuristics to optimize the overlay for delay, but these details are not relevant to our discussion. The reader is refered to [76] for further details.

One concern with the approach adopted by HMTP is that it could result in a tail in depth. The above example results in an increased depth for L and its descendants. We consider this further in Section 5.3.3.

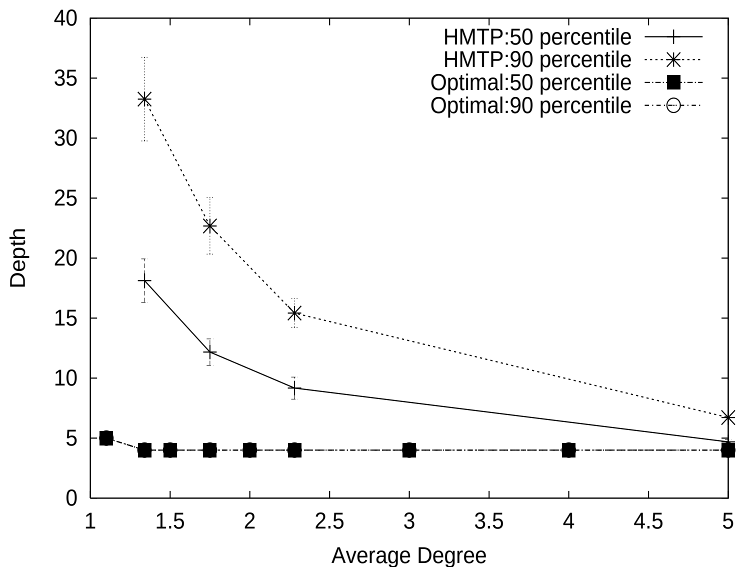

Figure 5.5: Depth Vs. Average Degree. The fraction of non-contributors is fixed at 65%.

5.3.2 Issues with heterogeneous constraints

There are two key concerns with heterogeneous node constraints:

• *Rejections:* The tree constructed by a protocol could attain sub-optimal configurations, as for example shown in Figure 5.2(b). Here, the system as a whole has sufficient bandwidth resources to enable connectivity to all nodes. However, the subtree rooted at the source is saturated with non-contributors, and the bandwidth resources of nodes in the disconnected subtrees remains unutilized. Nodes in the disconnected subtrees eventually are *rejected,* or forced to exit the multicast session.
• *High Depth:* An optimal configuration in terms of depth is one where the nodes that contribute the most (i.e. highest degree) form the highest levels, with lower degree nodes at lower levels. In the absence of mechanisms that explicitly favor construction of such trees, a protocol would produce trees of high depth such as shown in Figure 5.2(c), and consequently poorer performance as discussed in Section 5.2.2.

We evaluate these concerns using simulations with the HMTP protocol, for practical settings summarized in Figure 5.1. Further, we consider the sensitivity of our results to: (i) the *average degree* of the system; and (ii) the fraction of non-contributors in the system. Note that increasing the fraction of non-contributors while holding the average degree constant increases the variance (a measure of heterogeneity) in node degree.

Figure 5.4 plots the fraction of rejected participants as a function of the fraction of non-contributors. We assume that a rejection occurs if a host is unable to reconnect to the source within 30 seconds. For each constitution of participants (i.e., average degree and fraction of non-contributors), we plot the mean and standard deviation across 10 runs. Each curve corresponds to configurations with the same average degree. We notice two trends. First, as the fraction of non-contributors increases, the rejection rate increases. Second, for the same fraction of non-contributors, the rejection rate decreases as the average degree increases. Overall, the rejection rates are highest in regimes where there is a *combination* of constrained bandwidth environments (low average degree), and a high fraction of non-contributors. Further, the rejection rates are high under realistic settings of Table 5.1: over 10% in the *Slashdot* configuration, and over 40% in the *Gnutella* configuration. Finally, the standard deviations are high. This is because the occurance of rejections is sensitive to the particular run, and depends on the particular sequence of joins and leaves.

Figure 5.5 plots the depth as a function of the average degree, assuming that 65% of hosts are non-contributors. We compute the mean depth of the node by sampling its depth at different time instances in the session, and compute the 50 and 90 percentiles across all the nodes. We also plot the 50 and 90 percentiles of the optimal depth trees. As the average degree decreases, the depth of the HMTP trees increases. However, the depth of the optimal tree remains unaffected because even with a low average degree (close to 1), there are a sufficient number of high-degree (degree 10) nodes to ensure a low depth. We also considered varying the fraction of non-contributors while keeping the average degree fixed. While the optimal depth could decrease with a higher fraction of non-contributors, the constructed trees have depths that remain largely independent of the fraction of non-contributors.

5.3.3 Techniques for differential treatment

We considered the following key techniques for differential treatment which we incorporated into HMTP to address the concerns related to rejections and high depth raised in Section 5.3.2.

- *Preempt-Degree:* When a node, say A, joins the system, or has been disconnected because of a parent leave or preemption, it adopts the same *Depth-First-Search* procedure as described in the HMTP protocol. However, during its search for a parent, if it finds a node P having a child C with a lower degree than A, it can displace C and take its slot. The preempted node C treats the situation like a parent failure and does a DFS beginning at its grandparent. We note that C may not be directly able to choose A as a parent, because A might itself be saturated with existing children.
- *Preempt-Zero:* This technique is similar to *Preempt-Degree* except that only non-contributor hosts are preempted. In particular, nodes with degree 10 or 2 can preempt degree 0 nodes, but a node with degree 10 cannot preempt a node with degree 2.

In addition to these techniques, we consider a heuristic called *Preempt-Degree-Height,* which is motivated by the fact that even under homogeneous degree constraints, HMTP can result in imbalanced trees. For example, in Figure 5.2(a), if P fails, only one of the children (say C) can attach itself to the grandparent G. The other child L conducts a depth first search and connects to the first node with available capacity (say F). This results in an increased depth for L and its descendants. In the *Preempt-Degree-Height* heuristic L may preempt

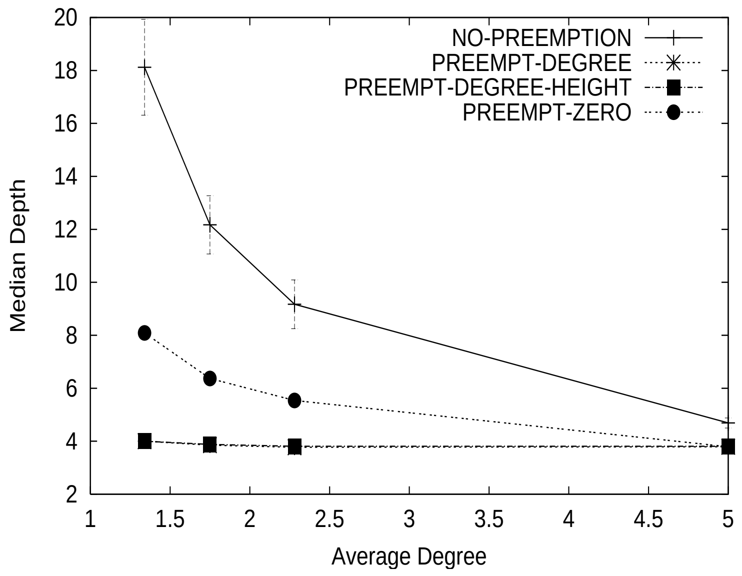

Figure 5.6: Depth Vs. Average Degree with 50% non-contributors.

a node that has a lower *height*, provided L has an equal or greater degree than that node. The height of a node is its distance to the farthest leaf in its subtree, and is maintained in a distributed fashion by having each node periodically send its height upstream.

While preemption can avoid sub-optimal configurations and help improve depth, it introduces additional disruptions to the tree structure. Of the heuristics above, intuitively we would expect *Preempt-Zero* to perform well as it completely eliminates rejections, and the only preemptions it causes are to non-contributor nodes who do not have any descendants. However, both *Preempt-Degree,* and *Preempt-Degree-Height* result in the disruption of entire sub-trees, and further *Preempt-Degree-Height* can cause $O(logN)$ preemptions in each search. Thus, to understand the trade-offs involved with the use of *Preempt-Degree* or *Preempt-Degree-Height* as compared to using *Preempt-Zero,* we focus our evaluation around the following questions:

• Are the benefits of preemption significant in terms of the improvement in metrics like rejection rates, loss and depth?
• Do the gains in stability from constructing lower depth trees offset the instability incurred by the preemptions themselves? Are there any benefits to adopting *Preempt-Degree* as compared to *Preempt-Zero* ?
• Does preemption improve the performance of the system as a whole, or does it penalize nodes in the lower degree classes?

5.3.4 Results

In this section, we evaluate the performance of *Preempt-Zero, Preempt-Degree,* and *Preempt-Degree-Height.* In all the graphs in this section, we present results for various

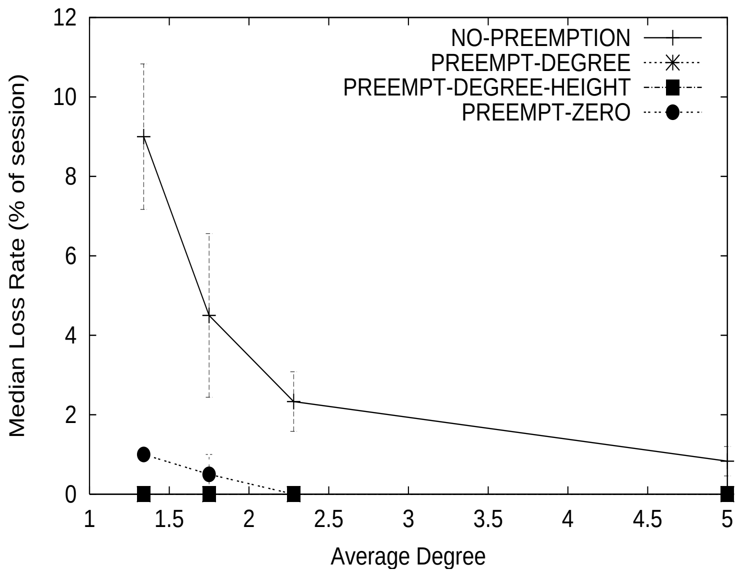

Figure 5.7: Loss Rate (% of session) Vs. average degree with 50% non-contributors.

values of average degree, but keep the fraction of non-contributors fixed at 50%. We discuss the impact of varying fraction of non-contributors in Section 5.3.5.

We first consider the depth metric. We compute the mean depth of the node by sampling its depth at different time instances in the session. We compute the median across all the nodes. Figure 5.6 plots the median as a function of average degree with 50% non-contributors. As expected, *Preempt-Degree* lowers the depth as compared to *No-Preemption.* Further, *Preempt-Degree* does better than *Preempt-Zero,* and the benefits are more prominent at lower values of average degree. Finally, *Preempt-Degree* seems to perform similar to *Preempt-Degree-Height* - this is because the latter heuristic improves only the tail in depth and not the median value.

We study the impact of preemption on the loss rates seen by the application. We consider a model where hosts gracefully leave, informing children about their departure, but not forwarding data in the interim time. This allows typical recovery times of the order of a few hundred milliseconds. We note however that the recovery times with *No-Preemption* can be significantly higher when sub-optimal configurations such as shown in Figure 5.2(b) are created. We compute the mean session loss seen by receivers, measured as the fraction of the session duration for which a node is disconnected from the source. We consider the median across receivers. Figure 5.7 plots the loss rates seen with various schemes. The loss rates with all preemption schemes is lower than *No-Preemption.* Further, at low average degrees, *Preempt-Degree* is more effective at reducing loss than *Preempt-Zero,* while *Preempt-Degree-Height* does not appear to improve loss much compared to *Preempt-Degree.* The reduction in loss with preemption is due to both a reduction in tree depth, and an elimination of longer recovery times that may occur with *No-Preemption* .

We have also conducted experiments with non-graceful leaves, which result in a higher recovery time. Here, the loss rates increase for all schemes, but the loss rate with *Preempt-*

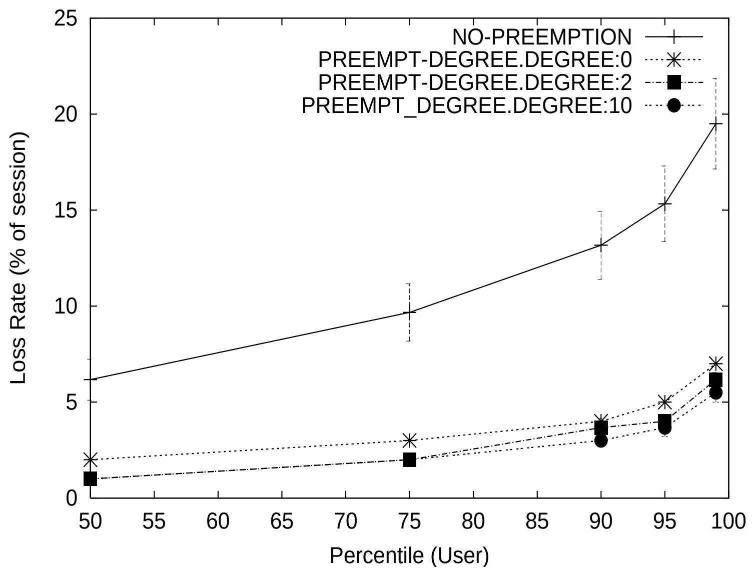

Figure 5.8: Loss rate distribution for different receiver classes. Avg. Degree = 1.34, 50% non-contributors.

Degree is much lower than other schemes. The exact magnitude depends on the fraction of leaves that are non-graceful and the time to detect the failure of a parent. While these experiments demonstrate the potential benefits to minimizing depth by employing *Preempt-Degree,* we expect the technique to complement rather than substitute other mechanisms to improve protocol resiliency such as redundancy [8, 14, 43] or error recovery [60].

It is particularly interesting that the loss rate with *Preempt-Degree* is lower than that with *Preempt-Zero*, despite the fact that *Preempt-Zero* only causes disruptions to non-contributor nodes that have no descendants, while *Preempt-Degree* can affect nodes with large subtrees. Our results show that the reduction in depth with *Preempt-Degree* and the consequent reduction in disruption due to ancestor departures more than offsets the disruption due to preemption itself. This benefit of preemption is likely to be even greater when there are other sources of disruption such as transient congestion on upstream links, or failure of ancestors.

We provide a possible informal explanation for why the disruptions caused by preemptions could be low. The disruption caused due to preemption of nodes of a particular class depends on: (i) the rate at which nodes of that class are preempted; and (ii) the number of affected descendants per preemption which in turn depends on how "high" in the tree the preempted node is located. The rate of preemption is likely to be high only if there is a substantial fraction of nodes of higher degree classes. But, the larger the fraction of nodes of higher degree classes, the lower the location of the preempted node in the tree. This is because preemption constructs trees with nodes of higher degree placed higher in the tree.

Finally, we consider the performance seen for different categories of receivers with *Preempt-Degree.* We compute the mean session loss seen by each receiver, and compute the median and higher percentiles across receivers. Figure 5.8 plots the distribution of

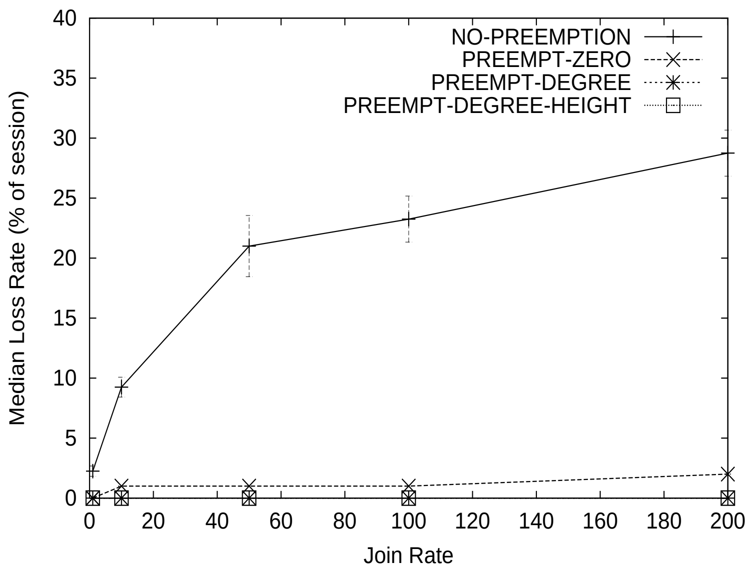

Figure 5.9: Impact of scale by varying join rate. Mean stay time fixed at 300. Average degree 1.34 and 50% non-contributors.

mean session loss across receivers, for a configuration with average degree 1.34, and 50% of participants being non-contributors. The top-most line corresponds to *No-Preemption.* Over 50% of the users see a loss above 10%, while 10% of the users see loss above 20%. The lower 3 curves corresponds to the loss rate distributions for the 3 classes of receivers of different degrees. While the lowest class of receivers does see worse performance than the higher classes, its performance is much better than the *No-Preemption* scheme. This indicates that preemption helps improve the performance of the system as a whole.

5.3.5 Sensitivity

We investigate key factors that affect our conclusions:

Effect of Scale: To study the scaling behavior, we keep the mean stay time fixed, and increase the join rate. This results in larger group sizes in steady state. We consider the mean session loss rates seen by receivers, and compute the median across receivers. Figure 5.9 plots the median as a function of the join rate (and hence group size). The loss rates without preemption increase sharply with group size. The loss rate with *Preempt-Zero* also slightly increases, but with *Preempt-Degree* and *Preempt-Degree-Height* losses grows very slowly.

Varying Constitution: Our results so far have been conducted by fixing the fraction of non-contributors at 50% and varying the average degree. We conducted experiments with a fixed average degree of 2.28, and varying the constitution. We consider the mean session loss rates seen by receivers, and compute the median across receivers. Figure 5.10 plots the median as a function of the fraction of non-contributors. We find that the benefits of using

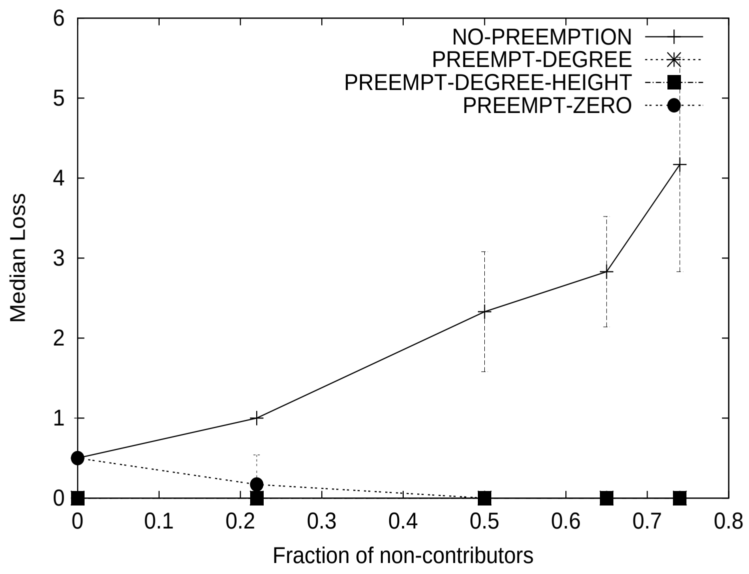

Figure 5.10: Impact of varying constitution with average degree fixed at 2.28.

Preempt-Degree as opposed to *Preempt-Zero* becomes more significant as the fraction of non-contributors (degree 0) nodes decreases. This is because in these regimes the fraction of lower degree nodes (degree 2) becomes large enough that it becomes important to ensure these nodes do not occupy the highest positions in the tree structure.

Heterogeneity Model: We considered how sensitive this result is to the ratio of the degrees between the larger and smaller contributors. We consider a simple two class model, fix the degree of the upper class at 10 and vary the degree of the lower class. While we omit detailed results, we find that *Preempt-Degree* always performs better than *No-Preemption* and the benefits are most significant in regimes where there is significant disparity in contribution levels of the nodes.

5.3.6 Summary

In this section, we showed that it is critical to incorporate techniques for differential treatment based on bandwidth constraints of nodes. Further, it is feasible to extend performance-centric protocols with heuristics for differential treatment of nodes. We introduced the *Preempt-Degree* technique in the HMTP protocol, and showed that it can eliminate rejections and reduce interrupt rates and loss. Our detailed simulation study demonstrates that *Preempt Degree* performs better than *Preempt-Zero* in environments where the overall fraction of lower degree nodes among all contributors is high, where the number of nodes in the system is large, and where the disparity in contribution levels is high. In such regimes, the benefits gained by constructing trees with lower depth using degree preemption outweigh the additional instability due to preemptions. This results in a net decrease in the loss rate. We also find that preemption improves the performance of even the worst class of receivers, and hence is beneficial to the system as a whole. While our results our promising, an actual design will need to consider other dimensions of heterogeneity such as the duration a node stays, and include heuristics to prevent preemption by nodes with high degree but poor geographical location. We defer these issues to future work.

5.4 DHT-based protocols

In this section, we consider the implications of heterogeneous bandwidth constraints for DHT-based protocols. All the issues pertaining to performance-centric protocols presented in Section 5.3 also apply to DHT-based protocols. However, in this section, we focus on additional issues unique to DHT-based approaches.

While there have been several DHT-based proposals for multicast in recent years [36, 49, 66, 37], we choose to focus on Scribe. Scribe is one of the more mature proposals among DHT-based approaches with well-defined mechanisms to honor per-node degree constraints. A more recent follow-up work SplitStream [8] builds on top of Scribe and considers data delivery along multiple trees, rather than a single tree to improve the resiliency of data delivery. While we draw on some of the extensions proposed in Splitstream, we only consider single tree data delivery in this chapter. We discuss some of the implications of multiple-tree solutions in Section 5.4.5.

5.4.1 Pastry/Scribe

Scribe [37] is a decentralized application-level multicast protocol built on top of the Pastry DHT protocol [55]. Pastry is targeted at settings which involve support of a large number of multicast groups. Each group may involve only a subset of the nodes in the Pastry system, but members in Pastry not part of a particular multicast group may be recruited to be forwarders in any Scribe tree. In this chapter however, our evaluation is conducted assuming all participating members in Pastry are also part of the Scribe tree.

Each node within Pastry is assigned a unique 128-bit `nodeId` which can be thought of as a sequence of digits in base 2^b (b is a Pastry parameter.) A Pastry node in a network of N nodes maintains a routing table containing about $\log_{2^b} N$ rows and 2^b columns. The entries in the r^{th} row of the routing table refer to nodes whose `nodeId`s share the first r digits with the local node's `nodeId`. The routing mechanism is a generalization of hypercube routing: each subsequent hop of the route to the destination shares longer and longer *prefixes* with the destination `nodeId`.

Scribe utilizes Pastry's routing mechanism to construct multicast trees in the following manner: each multicast group corresponds to a special ID called `topicId`. A multicast tree associated with the group is formed by the union of the Pastry routes from each group member to the `topicId`. Messages are multicast from the root to the members using reverse path forwarding [15].

A key issue with Scribe is that the number of children of a node A in the Scribe tree can be as high as the *in-degree* of the node in the underlying Pastry infrastructure – that is, the number of nodes in Pastry which use A as the next hop when routing towards the `topicId`. In general, this may be greater than the bandwidth constraints of the node. In order to tackle this overloading of nodes, the authors of Scribe/SplitStream have proposed two mechanisms:

• *Pushdown:* Whenever an overloaded node A receives a request from a potential child X, it can drop an existing child C, if X is found to be more "desirable" as a child than C. The orphaned node (either C or X) can contact one of the children of A as a potential parent, and this process goes on recursively. Choosing the criteria to determine which child of A (if any) that X should displace is an important issue. We discuss this in greater detail in Section 5.4.3.
• *Anycast.* If all nodes in the system have non-zero degree constraints, pushdown is guaranteed to terminate since leaf nodes will always have capacity. However, in the presence of non-contributor (degree 0) nodes, pushdown could end at a leaf that does not have capacity. The authors encountered this issue in the context of Splitstream [8]. This is tackled by an anycast procedure which provides an efficient way to locate a node with free capacity.

5.4.2 Issues with heterogeneous constraints

DHT-based approaches like Scribe need to deal with many of the issues that were raised with performance-centric protocols in Section 5.3, such as avoiding rejections and constructing low-depth trees. However, in addition, they need to address key concerns with regard to preserving the structure of the DHTs. In particular, while the pushdown and anycast operations described in Section 5.4.1 help Scribe deal with heterogeneous node bandwidth constraints, they may result in the creation of parent-child relationships which correspond to links that are not part of the underlying Pastry overlay. We term such links as *non-DHT* links. We believe these non-DHT links are undesirable because: (i) the route convergence and loop-free properties of DHT routing no longer apply if non-DHT links exist in significant numbers; and (ii) they require explicit per-tree maintenance which reduces the benefits of DHTs in terms of amortizing overlay maintenance costs over multiple multicast groups (and other applications).

Our evaluation with Scribe thus measures both the depth, and the fraction of non-DHT links in the trees constructed. While the depth captures performance of the tree, the fraction of non-DHT links measures the extent to which we need to deviate from the DHT structure while constructing Scribe trees and honoring node bandwidth constraints. A large fraction of non-DHT links diminishes the benefits of using Pastry.

5.4.3 Techniques Evaluated

We present two variants of the pushdown algorithm that we evaluated in Scribe. The first policy, *Preempt-ID-Pushdown* is based on the policy implemented in [8], and is not optimized to minimize depth in heterogeneous environments. The second policy, *Preempt-Degree-Pushdown*, is a new policy that we introduced in Scribe to improve depth in heterogeneous environments.

• *Preempt-ID-Pushdown:* When a saturated node A receives a request from a potential child X, X preempts a child C of A if X shares a longer prefix with the `topicID` than C. Further, the orphaned node (X or C) contacts a child of A and continues the pushdown if the orphaned node shares a prefix match with the child. However, if no child of A shares a prefix with the orphaned node, we continue with the pushdown operation by picking a random child of A.[2] An anycast operation is employed if a leaf node is reached without a parent being found.

• *Preempt-Degree-Pushdown:* Here, node degree rather than node id is the primary criterion in the pushdown. When a saturated node A receives a request from a potential child X, X preempts the child (say C) of A which has the lowest degree, provided X itself has a higher degree than C. The orphaned node (X or C) picks a random child of A that has a degree equal to or greater than itself and continues the pushdown. An anycast operation is employed if a leaf node is reached without a parent being found.

While *Preempt-Degree-Pushdown* can improve the depth of trees produced by Scribe compared to *Preempt-ID-Pushdown,* it can lead to the creation of a larger number of non-DHT links given that the id is no longer a key criterion in pushdown. Further, *Preempt-Degree-Pushdown* itself cannot create perfectly balanced trees - for example, if node A has a lower degree than node X, there is no mechanism in place for X to displace A. Doing so would require further deviation from the DHT-structure, and in particular would involve the creation of additional non-DHT links. In fact, we believe it is not easy to construct trees with both low depth, as well as a low fraction of non-DHT links. We discuss this further in Section 5.4.4.

5.4.4 Empirical Results

We present the results of experiments with Scribe with both homogeneous and heterogeneous degree constraints. We use the Scribe and Splitstream implementation [56] obtained from the authors for our experiments. We refer the reader to [62] for certain implementation issues that we found with the code-base and the precautions we took to ensure our results are not affected by implementation artifacts.

Homogeneous Environments: We assume that all nodes have a degree H. Figure 5.11 plots the fraction of non-DHT links within the Scribe tree as a function of H. Each curve corresponds to a different value of b, the base of the node IDs in Pastry. We find the fraction of non-DHT links is high and over 40% for all configurations we evaluate.

We now discuss factors that contribute to the creation of non-DHT links in Figure 5.11. Consider a `topicID` of 00...00. Let $\mathbf{0}*$ represent the nodes that have their first digit match the `topicID` (that is, the first digit is 0 and the rest of the digits are arbitrary). A join or reconnect request from any node in Scribe should be routed in the first hop to a $\mathbf{0}*$

[2]This is a slight departure from [8], where an anycast operation is employed if no child of A shares a prefix with the orphaned node. We have observed better performance in depth in homogeneous environments with our optimization. The intuition is that pushdown tends to do better at filling up nodes higher in the tree, while anycast tends to choose parents at more random locations in the tree.

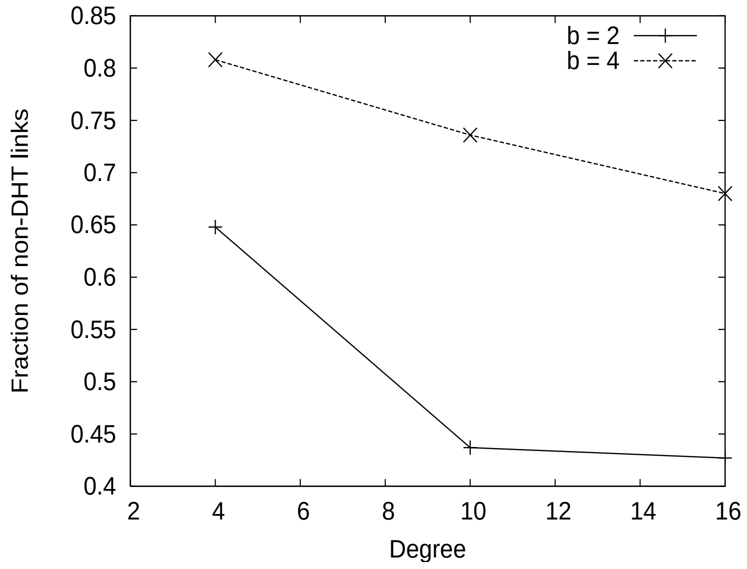

Figure 5.11: Fraction of non-DHT links (mean over the session) in homogenous environments for various values of node degree and b, the base of the node IDs in Pastry.

node, since we would like to match at least the first digit of the `topicID`. So, if there were no pushdown operations, given the reverse-path nature of tree construction in Scribe, all parents in a Scribe tree would be $\mathbf{0}*$ nodes.

A key factor leading to the creation of non-DHT links is that the total bandwidth resources at the $\mathbf{0}*$ nodes may not be sufficient to support all nodes in the tree. Let b be the base of the node IDs in Pastry, and AD be the average degree of the nodes in the system. Then, the $\mathbf{0}*$ nodes represent a fraction $\frac{1}{2^b}$ of the total nodes of the system, and we expect them to only be able to support a fraction $\frac{AD}{2^b}$ of the nodes in the system. Thus, we expect to see $1 - \frac{AD}{2^b}$ links that have non-$\mathbf{0}*$ nodes as parents. Such links are likely to be non-DHT links. This is because: (i) these links must have been created by pushdown operations as described above; and (ii) there are no explicit mechanisms in place to prefer choosing DHT links during a pushdown.

From this discussion, we expect the number of non-DHT links to be equal to $1 - \frac{H}{2^b}$ in a homogeneous environment, where all nodes have a degree H (as the average degree $AD = H$). While this partially explains Figure 5.11, the fraction of non-DHT links is significantly higher than our estimate. In particular, if $H \geq 2^b$, then we would not expect to see any non-DHT links. However, even when $H = 16$ and $b = 2$ so that $H \gg 2^b$, non-DHT links constitute over 40% of the links in the tree.

Further investigation revealed that the additional non-DHT links were created due to a sharp skew in the fan-ins of the $\mathbf{0}*$s in the system. The fan-in of a node is the number of other nodes in the system that have this node as a neighbor in Pastry. Due to the skew observed, Scribe join requests hit the $\mathbf{0}*$s non-uniformly, causing a much larger number of pushdowns, and hence non-DHT links. This also results in poor utilization of the available bandwidth resources at many of the $\mathbf{0}*$ nodes. We believe that the skew arises due to Pas-

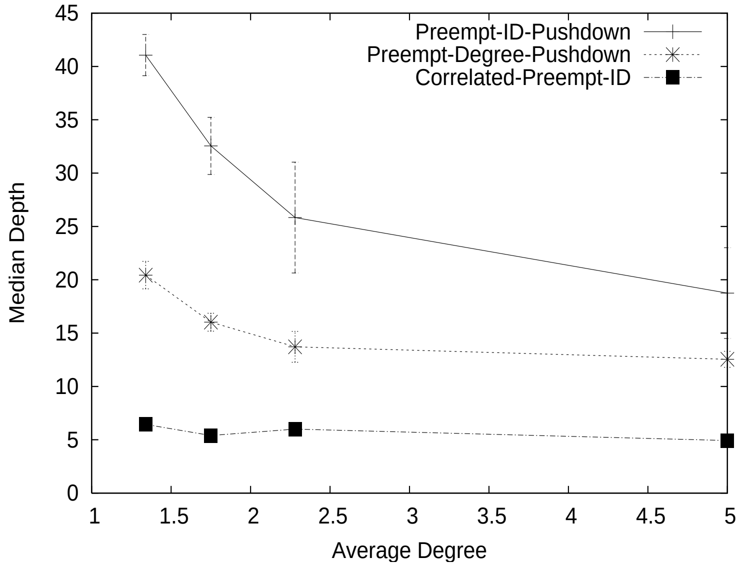

Figure 5.12: Depth Vs. Average Degree in heterogeneous settings. We compute mean depth of a node during the session, and compute median across the nodes. The fraction of non-contributors is fixed at 50%.

try's join and repair mechanisms in which a new node picks up routing table entries from other nodes in the system, which makes nodes that joined earlier far more likely to be picked as neighbors as compared to other nodes. We defer to future work an examination of how fundamental the skew is to the design of Pastry/Scribe, and whether it can be eliminated using simple heuristics.

Heterogeneous Environments: Figure 5.12 compares the depth of the Scribe multicast tree created with the *Preempt-ID-Pushdown* and *Preempt-Degree-Pushdown* in heterogeneous environments. The experiments assume 50% of the nodes are non-contributors (degree 0), and are repeated for various average degree values. The optimal median depth for any of the plotted configurations (not shown in the graph) is about 4. The top 2 curves correspond to *Preempt-ID-Pushdown* and *Preempt-Degree-Pushdown. Preempt-ID-Pushdown* performs significantly worse than optimal, which is as expected given that there are no mechanisms in place that optimize depth in heterogeneous environments. *Preempt-Degree-Pushdown* performs better than *Preempt-ID-Pushdown* but is still not optimal. This is consistent with our discussion in Section 5.4.3.

Figure 5.13 shows the fraction of non-DHT links from our simulations for *Preempt-Degree-Pushdown,* and *Preempt-ID-Pushdown.* The fraction of non-DHT links is over 80% for a range of average degrees. We believe both factors that we discussed with homogeneous environments – insufficient resources at $\mathbf{0}*$ nodes, and the skew in the in-degree of Pastry – have contributed to the creation of non-DHT links. Further, as discussed above, even if the skew could be completely eliminated, we would still expect to see $1 - \frac{AD}{2^b}$ non-DHT links due to insufficient resources at $\mathbf{0}*$ nodes, where AD is the average degree of the nodes in

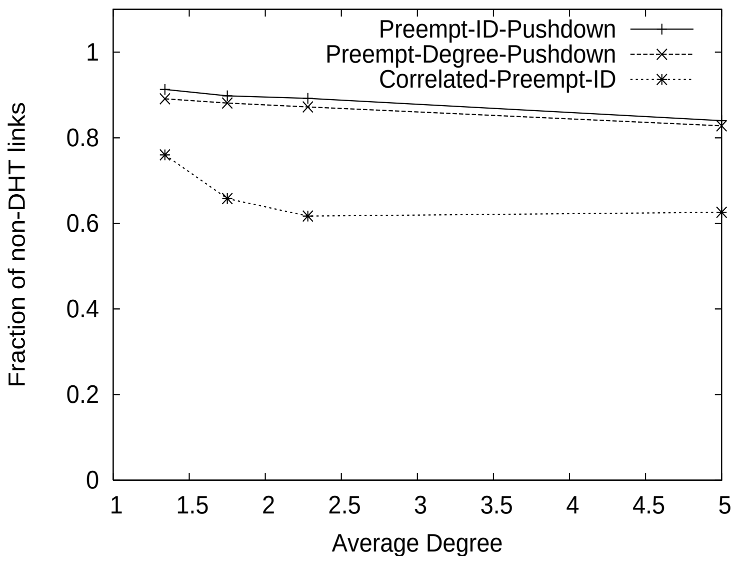

Figure 5.13: Fraction of non-DHT links Vs. Average Degree in heterogeneous settings. The fraction of non-contributors is fixed at 50%.

the system.

In addition to these factors, policies that optimize the depth of trees in Scribe in heterogeneous environments may potentially result in an even higher fraction of non-DHT links. For example, in an environment with nodes of degree H, L, and 0 ($H > L$), the optimal depth tree requires having all nodes of degree H at the highest levels in the tree, and thus as interior nodes. However, only a fraction $\frac{1}{2^b}$ of nodes of degree H are likely to be `0*` nodes and using the other nodes would likely result in non-DHT links. This leads us to expect that *Preempt-Degree-Pushdown* would have a higher fraction of non-DHT links as compared to *Preempt-ID-Pushdown*, given that it results in better depth trees. However, both policies perform similarly (Figure 5.13). We believe this is because the other factors causing non-DHT links dominate in our experiments resulting in an already large fraction of non-DHT links for *Preempt-ID-Pushdown*.

5.4.5 Discussion of Potential Solutions

We considered the impact of heterogeneous bandwidth constraints on DHT protocols. Our evaluation focused on Scribe, one of the more mature DHT-based proposals for overlay multicast, which has clear and well-defined mechanisms for dealing with per-node constraints. These mechanisms can result in the creation of *non-DHT* links in the Scribe tree, that is, links which are present in Scribe but not in Pastry. We believe that it is important to ensure a low fraction of non-DHT links in order to leverage the benefits of DHTs (loop avoidance and id-based routing, amortizing overhead involved in constructing multiple trees). Our experiments with Scribe reveal a significant fraction of non-DHT links.

There are three factors that can cause this result. First, the bandwidth resources of nodes that share a prefix with the `topicId` may not be sufficient to sustain all nodes in the system. Second, minimizing depth of trees in Scribe requires utilizing higher degree nodes, even though they may not share a prefix with the `topicId`. The third factor is a skew in the in-degree of Pastry. We believe the skew is a result of specific heuristics employed in Pastry, and can potentially be minimized. However, we believe the first two factors are fundamental to the mismatch of node bandwidth constraints and node ids with DHT-based designs. While we do not breakdown the contribution of each factor in our experiments, simple analysis shows that the first factor alone could lead to the creation of $1 - \frac{AD}{2^b}$ non-DHT links, where AD is the average degree of the system, and b is the base of the node IDs in Pastry. In the rest of the section, we discuss potential ways of addressing the issue.

ID-Degree Correlation: A natural question is whether changing the random id assignment of DHTs, and instead employing an assignment where node ids are correlated to node bandwidth constraints can address the issue. To evaluate the potential of such techniques, we consider the *Correlated-Preempt-ID* heuristic, where nodes with higher degrees are assigned `nodeId`s which share longer prefixes with the `topicId`. Figure 5.12 shows that this policy indeed is able to achieve depths close to the optimal depth of 4, while Figure 5.13 shows it can significantly lower the fraction of non-DHT links. However, while such a solution could work in scenarios where the DHT is primarily used for a specific multicast group, disturbing the uniform distribution of DHT `nodeId`s can be undesirable. Further, DHT's are particularly useful in scenarios where there is a shared infrastructure for a wide variety of applications including multicast sessions. In such scenarios, it is difficult to achieve a correlation between node id and node degree assignments across all trees.
Multiple Trees: Another question is whether the issues involved can be tackled using the multi-tree data delivery framework [8, 43] employed in SplitStream to improve the resiliency of data delivery. In this framework, 2^b trees are constructed, with the `topicIds` of every tree beginning with a different digit. Each node is an interior node in the one tree where it shares a prefix with the `topicId`, and is a leaf node in the rest. We note that a direct application of the multi-tree approach cannot solve the problem - if nodes belong to multiple degree classes to begin with, then, each of the trees will continue to have nodes of multiple degree classes, and the issues presented in this chapter continue to be a concern.
Multiple Trees with Virtual Servers [19]: One potential practical direction for solving the issues with DHTs is to combine the multi-tree data delivery framework with the concept of virtual servers proposed in [19]. The idea here is that a node can acquire a number of ids proportional to its degree, and then use the multi-tree data delivery framework above. The trade-off then is that we are not completely concentrating the resources of a higher degree node in one tree, rather, we are distributing it across several trees, thereby giving up on the policy of interior disjointness. We believe this may be a promising direction, and the implications of this trade-off would be interesting to evaluate.

5.5 Summary and Conclusions

Our experience with a broadcasting system (Chapter 4) highlighted the importance of leveraging the heterogeneity in node capabilities while deploying ESM in environments where

the overall bandwidth resources in the system are scarce. In this chapter, we systematically consider the impact of heterogeneous outgoing bandwidth capabilities of nodes on protocol designs for overlay multicast. We consider the implications of this issue on representative examples from two classes of protocol designs – performance-centric and DHT-based.
Our key contributions are:

• We demonstrate that differential treatment of nodes based on their bandwidth capabilities must be a first-order criterion in protocol design. We show that techniques for differential treatment can be incorporated in performance-centric protocols, and demonstrate their benefits in reducing depth, and overall losses using a detailed simulation study. To our knowledge this is the first systematic analysis of the trade-offs involved with an approach based on differential treatment.
• We show that the id-based structure of DHT-based protocols complicates differential treatment. We consider the fraction of *non-DHT links* to help quantify the trade-offs. Our analysis indicates that the fraction of non-DHT links could be significant due to the mismatch between the id space that underlies the DHT structure and node bandwidth constraints. We evaluated and discussed some possible ways of improving the design.

Our future work involves evaluating techniques for differential treatment in actual Internet implementations of overlay protocols, and considering additional dimensions of heterogeneity such as the duration that nodes stay. We also hope to further explore the trade-offs between performance-centric and DHT-based protocols.

Chapter 6

Related Work

End System Multicast was the first published proposal that argued for and demonstrated the viability of an overlay approach to multicast [11]. Since our initial proposal [11], there has been a significant interest in architectures and protocols for overlay multicast. The field of overlay multicast has emerged as an important one of its own right. In this chapter, we discuss some of the advances in the field.

Mirroring the three broad areas in which this thesis makes contributions, we will organize our discussion by covering related work in:

- Overlay Multicast Architectures
- Protocol Designs for Overlay Multicast
- Broadcasting/conferencing systems

While there have been works that have investigated one or two of the above broad dimensions, this thesis is unique in that it makes contributions to all three of the dimensions. Further, it has adopted an **integrated** approach to architecture, protocol design and system building. The primary goal of validating the End System Multicast architecture has motivated us to build a system, and this in turn has influenced an empirical approach to protocol design.

6.1 Architectures for Multicast

The research community has been cognizant of issues with the IP Multicast architecture. In this section, we discuss other work that has proposed alternate architectures for multicast.

Application end-point based overlays: Yoid [24] and ALMI [45] are the architectural proposals closest in spirit to End System Multicast. Both these efforts were conceived independently of End System Multicast in work concurrent to our own [11]. Both proposals have explored pushing multicast functionality to end systems participating in the group, and propose protocols for constructing overlay structures. While this thesis shares the architectural vision of these proposals, it has since then gone much further to validating the vision, by pursuing a systems approach to protocol design. This has led the thesis to explore issues such as bandwidth adaptation, heterogeneity of nodes and ultimately to a fully operational and extensively deployed broadcasting system based on ESM.

Infrastructure-based Overlays Scattercast [10] and Overcast [34] also looked at an overlay approach to multicast. However, they have chosen to investigate another end of the architectural spectrum, *infrastructure-centric* architectures. Here, an organization that provides value-added Internet services deploys powerful hosts on the Internet at strategic locations, and end systems merely receive data from these hosts. Both Scattercast and Overcast provide protocols that can be used to organize hosts of service providers into efficient overlays. Such an approach is also being employed by commercial providers such as Akamai [2], Fast-Forward [30], and Real Broadcasting Networks [50].

The key reason we are motivated to explore an application end-point architecture is the goal of *ubiquitous* deployment of broadcasting and conferencing applications. An application end-point architecture is instantaneous to deploy, and one can set up a broadcast with minimal set-up overhead and low cost. Further, such an architecture is completely distributed, and leverages the bandwidth resources of end systems actually participating in the group. It can thus scale to support a very large number of multicast groups. In contrast, while an infrastructure service can potentially deal with a smaller number of well-defined groups, it is unclear whether it can support millions of groups associated with ubiquitous deployment.

While application end-point architectures have the promise to enable ubiquitous deployment, infrastructure-centric architecture can potentially provide more robust data delivery. Infrastructure architectures involve better provisioned proxy hosts that can be located at strategic locations on the Internet. Further, the probability of failure of such hosts is low. In contrast, application end-point architectures potentially involve a wider range of end systems that may not provide as good performance, and must deal with the failures of end systems and the dynamic join/leave behavior of such hosts. Finally, infrastructure-based architectures may scale better to very large group sizes given the better robustness properties they provide.

Other overlay multicast architectures: Application end-point, and infrastructure-based architectures represent two ends of a rich spectrum. However, there are several other possible architectural choices in this space. One interesting possibility for the longer term is waypoint-based architectures that we mentioned in Chapter 4. These architectures do not depend on pre-provisioned infrastructure nodes, but can dynamically invoke and leverage resources of waypoints to gracefully improve performance. One other hybrid architecture that has recently been explored is I3 [31]. This architecture relies on infrastructure nodes for the data path, but allows application end-points to control the tree construction process. Finally, there has been work that has argued for combining IP Multicast and an overlay approach to multicast [76]. Here, a multicast group consists of various multicast islands, each with designated members. An overlay protocol is used to connect designated members, and IP Multicast is used within each domain.

Refinements to IP Multicast framework: There have been proposals that have attempted to alleviate some of the concerns with IP Multicast but still retain the basic ideas of the IP Multicast architecture. Express [29] investigated a single-source multicast model where only a designated source can multicast to a group. The work also provided mechanisms for access control and charging. Reunite [68] investigated mechanisms for reducing the forwarding state in routers by requiring only routers that split and forward data along multiple interfaces to maintain per-group state. However, none of these works address fundamental concerns with IP Multicast - violation of the stateless architectural principles of Internet design, and support for higher level functionality such as congestion control and reliability.

MBone: The MBone [7] was a static and manually configured overlay, which was not meant as an alternative to IP Multicast, rather aimed at enabling the incremental deployment of IP Multicast. Organizations join the MBone by configuring a machine as an MBone router, which in turn is manually configured to an upstream router using an overlay tunnel. The key technical aspects that distinguish the MBone from overlay multicast proposals is the self-configuring and performance-aware aspect of the overlay construction, and the construction of the overlay among participating end systems without involving any other dedicated hosts.

6.2 Protocol Design

Narada was the first published protocol for overlay multicast. There were other independently conceived protocol designs published shortly after Narada such as [24, 34, 10, 45]. Since these early works, there have been several newer protocols in the community that have investigated different aspects of the design space. In particular, there has been significant work on algorithms for scalable group management, and improving the resiliency of data delivery through the construction of richer data-delivery structures rather than trees.

The thesis research may be distinguished from this body of work by the *systems,* or *empirical,* approach to protocol design. The protocol designs have been driven by the objective of being deployed in operational systems involving real applications and real users. This has led us to a focus on an orthogonal and complementary set of issues such as adaption to dynamic network metrics, issues pertaining to adaptation to available bandwidth, and issues related to the heterogeneity in node capabilities.

In the rest of this section, we discuss some of the advances in protocol design, and organize the discussion around the particular issues handled in the work.

6.2.1 Scalable Group Management

The Narada protocol was targeted at smaller-scale multi-source conferencing applications, and adopted a two-step approach to constructing data delivery trees. In the first step, it constructs a mesh among participating members, and in the second step, it constructs spanning trees of the mesh. A key issue with adopting Narada to larger group sizes is that it requires members to maintain complete membership state about all other members. Several protocol designs have since evolved in the community that were driven by the focus of constructing larger-scale overlays.

Yoid [24] was one of the earliest protocols that had more scalable group management algorithms. In contrast to the two-step mesh-based approach of Narada, Yoid construct trees directly, in that members choose parents from other members they know. Yoid adopted a shared tree approach, where a single overlay tree is constructed for all sources. A key concern with the shared tree approach however is that it is difficult to maintain connectivity of the tree structure. A shared tree approach involves the need to elect a root of the shared tree - handling failure of the root is not easy and it is difficult to analyze the robustness properties of the resulting designs. One potential way to simplify the design is to assume the existence of stable nodes (rendezvous points) that are present throughout the broadcast.

Overcast [34] was another early protocol that constructed trees directly, and which required members to know only a subset of group members. In contrast to Yoid however, Overcast was explicitly targeted at single source broadcasting applications. Given that the broadcast is relevant only as long as the source is alive, Overcast does not need to deal with concerns pertaining to failure of the core, and direct approaches to constructing trees are a natural solution in such cases.

Since these early works, there have been other protocols that have employed scalable group management algorithms [76, 28, 57, 33]. NICE [57] was one of the more representative of these protocols, and was targeted at large-scale and low latency (rather than high bandwidth) applications. NICE organizes receivers into a set of layers. Hosts in each layer are partitioned into a set of clusters, each of which has a size between k and $3k - 1$, where k is a protocol parameter. Each cluster elects a head, and the heads form the next layer. NICE was perhaps one of the earliest works to systematically evaluate the scaling properties of the algorithm. Detailed performance comparisons with Narada demonstrate that NICE [57] achieves competitive performance in metrics likes stress and latency, and significantly reduces control overhead for larger group sizes. However, NICE has similar concerns as with shared tree approaches like Yoid - it is difficult to analyze the robustness of the design in the presence of failures of the root of the hierarchy. Another important concern with NICE is that nodes higher in the hierarchy are required to support a greater number of children in a manner that does not consider their outgoing bandwidth constraints. This issue is partially addressed by modifications to the NICE algorithm in Zigzag [17].

Among other protocol designs, [33] constructs a 2-level hierarchy, and employs a variant of the Narada protocol on each level of the hierarchy. HMTP [76] is built around a depth-first-search like algorithm. Nodes that join conduct a search in this style to locate a parent with free capacity. When a parent dies, the children contact the grand-parent and again conduct a similar depth-first-search algorithm.

The Sparta protocol (Chapter 4) too employs scalable algorithms for group management, and draws on some of the work discussed above. It too uses a direct tree-construction algorithm, given that it is targeted at single-source broadcasting applications, and does not need to deal with robustness concerns of shared-tree approaches. In our broadcasting protocol, members maintain information about a random set of hosts that are uncorrelated to the tree, in addition to path information. This is in contrast to protocols like Overcast [34] and NICE [57], where group membership state is tightly coupled to the existing tree structure: Yoid [24] also maintains group membership information decoupled from the tree structure - however the mechanisms they adopt are different. Sparta uses a gossip protocol adapted from [52], while Yoid builds a separate random control structure called the mesh, and Scribe constructs a topology based on logical identifiers.

Finally, in contrast to all these proposals, ALMI [45], advocates a centralized approach to overlay construction. Here, a central entity (typically the source) maintains knowledge of the entire group. When a node needs to look for a new parent, it requests the source and who assigns it an appropriate candidate. While centralized approaches can simplify coordination in tree construction, and can greatly simplify design, a natural concern is the control overheads with large group sizes, and robustness concerns with failures close to the central server.

6.2.2 DHT-Based Approaches

In all the protocols we have described so far, the focus is on constructing overlay topologies where neighbor relationships are primarily governed by performance. Such protocols typically incorporate special group management mechanisms to maintain knowledge of a subset of group members. When a parent dies, or bad performance is observed, nodes switch to a parent they believe can provide better performance. Nodes may also perform periodic probes for locating better parents. We term such protocols as being *performance-centric.*

In contrast to these approaches, a whole new class of approaches for overlay multicast which include designs such as Delaunay Triangulations [36], CAN-multicast [49], Bayeux [66] and Chord [67] have emerged. We term these protocols as *DHT-based* protocols. In these protocols, members are assigned addresses from an abstract coordinate space such as a ring, torus or hypercube. By creating neighbor links based on these addresses, a structured overlay is created which enables scalable and efficient unicast routing based on the node identifiers. These unicast routes are used for creating multicast distribution trees (using flooding or reverse path forwarding.)

DHT-based approaches originally evolved in the context of the design of large-scale content location systems such as Gnutella [25]. Since then, they have been viewed as providing attractive properties that can help in the design of scalable group management algorithms for overlay multicast. Two principal reasons have been advocated for a DHT-based approach. In the last few-years, advocates of DHT-based approaches believe that the approach provides a generic primitive that can benefit a wide range of applications including overlay multicast, and file-sharing. A second argument advanced for these approaches is that the same DHT-based overlay can be used to simultaneously support and maintain a large number of overlay applications and multicast trees. This could help keep the costs low as compared to constructing and maintaining many individual overlays.

A natural concern with DHT-based approaches is performance, given that selection of overlay links is not dictated by performance alone. Studying the performance potential of

DHT-based overlays, and mechanisms to enable good performance with DHTs is an active and ongoing area of research. Much of the work to date has focused on delay-based metrics. However, several other dimensions that affect protocol design including network dynamics, heterogeneity in host characteristics, and issues related to NATs and firewalls are yet to be considered.

6.2.3 Richer structures for data delivery

Another dimension in which we can taxonomize protocols for Overlay Multicast is on the structure used to deliver data to receipients. All the protocols in this thesis disseminate data using tree like structures. Trees have also been the standard structure for data delivery in a wide range of protocols [24, 11, 34, 10, 36, 66, 37, 57, 73]. Trees are the most natural way of delivering data, and in the absence of specialized coding algorithms, do not incur any additional redundancy. However, they suffer from the concern that failure of a member can affect a large number of other members in the overlay. To improve the overall resiliency of data delivery, recent years has seen the emergence of more redundant structures for data delivery.

CoopNet [43] and Splitstream [8] have investigated delivering data using multiple trees, rather than a single tree. In these protocols, the source uses a custom codec to encode the multimedia stream into many sub-streams using multiple description coding, and distributes each sub-stream along a particular overlay tree. The quality experienced by a receiver depends on the number of substreams that it receives. Further, each node is an interior node in exactly one tree, so that the failure of a node only affects its descendants in a given tree. This approach has several attractions. First, it improves the overall resiliency of the system. Second, it enables support for heterogeneous hosts by having each receiver subscribe to as many layers as its capacity allows. Third, such an approach may help make application end-point architectures viable in more extreme environments where all nodes are behind limited bandwidth connections like DSL and cable modems. For example, consider a scenario where the source rate is 100 Kbps, and all participating hosts have an out-going bandwidth of 100 Kbps. A single tree solution will result in a linear chain of participating hosts. In contrast, a multiple tree solution can result in the construction of two trees, with each node being an interior node in one tree, and capable of supporting two children in that tree.

Bullet [14] is another approach for improving the resiliency of data delivery where a mesh is constructed among participating receivers. Individual receivers are responsible for locating and retrieving data from multiple points in parallel, and there are mechanisms present which ensure data received from multiple parents is disjoint. However, it is not clear how effective these mechanisms are in suppressing duplicate packets. Further, the interaction between the overlay construction and coding algorithms have not been discussed.

6.2.4 Protocol Taxonomy

Figure 6.1 summarizes the discussion in this section, by presenting a taxonomy of the protocol designs. At the highest level protocols can be classifed as centralized or distributed,

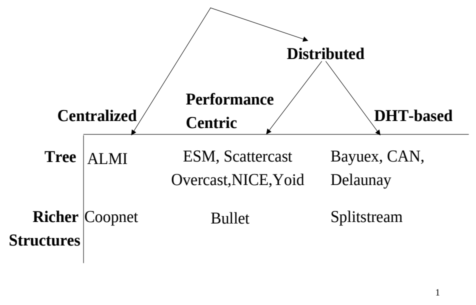

Figure 6.1: Taxonomy of protocol proposals

depending on whether the algorithms for group management are run in a distributed fashion or centralized at the source. ALMI [45] and Coopnet [43] represent such centralized protocols. Distributed protocols can be further classified as performance-centric or DHT-based, depending on the primary criterion in parent selection. Bayeux [66], CAN [48] and Scribe [37] are some of the DHT-based protocols. Among the performance-centric protocols, Narada and Gossamer are targeted at multi-source applications, and adopt a 2-step mesh-based approach for constructing overlays. Yoid [24] employs a shared-tree approach, while Overcast [34] and the protocol in the ESM broadcasting system are explicitly targeted at single-source broadcasting applications. Figure 6.1 also shows that an orthogonal dimension in which we can taxonomize protocols is based on the structure used for data delivery. While most protocols discussed above adopt tree-based structures, Coopnet [43] and SplitStream [8] adopt multi-tree based approaches, and Bullet [14] employs a mesh-based approach.

6.3 Broadcasting Systems

To our knowledge, the ESM broadcasting system is the only operationally deployed system in the research community that uses an application end-point based overlay multicast architecture. Recently, several peer-to-peer broadcast systems have been built by commercial entities [4, 9, 69] and non-profit organizations[44]. To our knowledge, many of these systems focus on audio applications which have lower bandwidth requirements. However, given the limited information on these systems, we are unable to do a detailed comparison.

Commercial entities, such as Akamai [2] and Real Broadcast Network [50], already provide Internet broadcasting as a charged service. They rely on dedicated, well-provision infrastructure nodes to replicate video streams. Such an approach has some fundamental advantages such as security and stable performance. However, these systems are viable only for larger-scale publishers, rather than the wide-range of low budget Internet broadcasting applications we seek to enable.

The MBone [7] Project, and its associated applications such as vic [39], vat [32], and MASH [38] made a great effort to achieve ubiquitous Internet broadcasting. However, the MBone could only touch a small fraction of Internet users (mostly networking researchers) due to the fundamental limitations of IP Multicast and dependence on the special MBone infrastructure. In contrast, our system has over a short time already reached a wide range of users, including home users behind a range of access technologies, and users behind NATs and firewalls.

Chapter 7

Conclusions and Future work

In this chapter, we summarize our key contributions, and present new research directions that are motivated by this thesis.

7.1 Contributions

Enabling the ubiquitous deployment of broadcasting and conferencing applications on the Internet has been a long-standing vision in the networking community. For much of the 1990's, the conventional wisdom was that such applications are best supported using the IP Multicast architecture. However fundamental challenges like per-group control state in routers, and support for higher level functionality like congestion control and reliability, have prevented IP Multicast from being widely deployed. Migrating multicast functionality from routers to end systems has the potential to address the issue - however a key challenge is performance.

We summarize the key contributions and distinctive highlights of this thesis below:

Architecture: The primary contribution of this thesis is demonstrating that multicast functionality can be efficiently supported without router support. We demonstrated this thesis in the context of the *End System Multicast* architecture. The thesis has been validated by extensive evaluation using simulations, Internet test-beds, and the deployment of an ESM-based broadcasting system that has been used by several thousand users.

Protocol Design: This thesis has made important contributions to the design of self-organizing protocols for overlay multicast. The scale of nodes involved, the dynamics of participation (group dynamics and Internet congestion), and the heterogeneity in the Internet make the design of these protocols very different than traditional distributed algorithms. The specific contributions of the thesis include:

- The design and implementation of **Narada,** perhaps the first self-organizing protocol for overlay multicast, the design of which was motivated by multi-source conferencing applications (Chapter 2).

- The design and implementation of **Sparta,** a protocol for single-source broadcasting applications that forms part of the operationally deployed broadcasting system. This is perhaps the only self-organizing protocol in the community that has seen real user experience (Chapter 4).

Empirical Approach to Protocol Design: The protocol designs in this thesis have been motivated by real applications, and by issues that arose while deploying such applications in real Internet settings using ESM. The thesis has identified and highlighted such issues, considered techniques to address them, and demonstrated the effectiveness of the techniques using systematic empirical evaluations. We summarize some specific contributions below:

• The thesis demonstrated that it is critical for self-organizing protocols targeted at conferencing applications to dynamically adapt to both bandwidth and latency. We presented techniques to achieve this, and showed how they could be incorporated in the Narada protocol. A detailed evaluation on a wide-area Internet test-bed demonstrated the heuristics worked well. We also considered light-weight probing techniques, and showed that techniques based on round trip time and bottleneck bandwidth are effective in guiding parent selection(Chapter 3).
• Experience implementing a protocol for a real broadcasting system has provided further insight into issues associated with bandwidth adaptation. Our experience reveals the need to consider loss rates to deal with VBR streams, and the need for dynamic tuning of the detection time (time taken to detect poor application performance) to the resources available in the environment and network connectivity at clients. While the performance with the ESM broadcasting system was satisfactory on the whole, the experience highlighted challenges with resource-constrained regimes. The experience also indicated the potential to meet these challenges through design refinements involving exploiting heterogeneity in node capabilities through differential treatment, and NAT-aware overlay construction (Chapter 4).
• The deployment experience led to a systematic study of the impact of heterogeneous out-going bandwidth contraints of nodes on existing protocols of two different classes for End System Multicast. We showed that differential treatment of nodes based on their bandwidth capabilities must be a first-order criterion in protocol design. We showed that techniques for differential treatment can be incorporated in performance-centric protocols, and demonstrated their benefits using a detailed simulation study. With DHT-based approaches however, straight-forward extensions are not sufficient to address the issue, owing to the mismatch between the ID space that underlies the DHT and the outgoing bandwidth constraints on nodes. (Chapter 5).

Influential role in new research area: This thesis has influenced significant follow-up work in the research community since the initial proposal of End System Multicast [11]. Overlay multicast has evolved into a field of its own right, as demonstrated by a rich body of work [24, 11, 34, 10, 36, 49, 66, 37, 57, 43, 73, 14, 8] that has emerged. The Narada protocol has been extensively used as a benchmark for comparison (for example, [57]). Metrics such as *Stress* and *Relative Delay Penalty* introduced by this thesis in [11] have been extensively used in the research community to evaluate overlay protocols.

System Artifacts: The self-organizing protocols developed in this thesis have been implemented and deployed in a fully operational broadcasting system based on End System Multicast. The system is satisfying the needs of real content publishers and viewers. It has been used to broadcast tens of events, and has been used by several thousand users. The extensive and sustained usage of the system are proving it to be an important contribution in its own right.

7.2 Future Work

We present research directions that are opened by this thesis research. The research directions include: (i) directions related to the viability of the End System Multicast architecture; (ii) hybrid architectures combining the ideas of End System Multicast with infrastructure-based architectures; (iii) issues related to protocol design; and (iv) issues related to the design of the ESM system.

7.2.1 Bandwidth Constrained Environments

Our experience with deploying the ESM broadcasting system revealed the challenges involved in deploying ESM in environments with a large fraction of hosts behind access technologies like cable modem and DSL, that have limited outgoing access bandwidth. In the long run, it is likely that access technology bandwidth will improve with higher speed connections to the home. However, in the foreseeable future, it is important to design solutions that can work in these environments.

One key direction is application level adaptation. For example, recent work [43, 8] have proposed a solution that involves delivering data using multiple trees, rather than a single tree. The source uses a custom codec to encode the multimedia stream into many sub-streams using multiple description coding, and distributes each sub-stream along a particular overlay tree. The quality experienced by a receiver depends on the number of substreams that it receives. Each node is an interior node in exactly one tree, so that the failure of a node only affects its descendants in a given tree. While originally proposed to improve resilience of data delivery, such an approach may help make application end-point architectures viable in more extreme environments where all nodes are behind limited bandwidth connections like DSL and cable modems. For example, consider a scenario where the source rate is 100 Kbps, and all participating hosts have an out-going bandwidth of 100 Kbps. A single tree solution will result in a linear chain of participating hosts. In contrast, a multiple tree solution can result in the construction of two trees, with each node being an interior node in one tree, and capable of supporting two children in that tree.

We believe the multi-tree solution represents just one point in a rich design spectrum that involves interplay between video coding and overlay construction, and many more interesting solutions may appear if we investigate further.

7.2.2 User Cooperation and Security Issues

In this thesis, we have focused on the performance concerns with ESM. However, there are other factors besides performance that may affect the viability of End System Multicast. We consider two such issues below:

• *User Cooperation:* ESM relies on users co-operating with each other to share their outgoing bandwidth resources, and it is unclear how willing they would be do so. Further, in environments with heterogeneous node capabilities, achieving good performance with ESM potentially requires unequal contributions from different participants. The questions are even more pertinent in environments where users pay based on their bandwidth usage as opposed to a flat Internet connectivity fee. To an extent, our system deployment has helped us gauge user behavior, and we find there is significant willingness among users to cooperate. One important direction for future research is to look at policies that provide explicit incentives for users to contribute more resources e.g. better quality to users who contribute more through differential treatment). From a system perspective, it would be interesting to explore APIs that enable users to customize and control the bandwidth contributed by them

• *Security:* The protocols proposed in this thesis are entirely dependent on cooperative and non-malicious behavior by participating end systems. In larger-scale real world deployments, it would be critical to systematically consider the behavior of the protocol in the presence of malicious hosts. An issue that is potentially easy to tackle is authenticating content from the source which can prevent an intermediate node from replacing original content with false content. However, a harder issue involves hosts that do not co-operate on the control path. For example, consider a gossip-based group management algorithm where a host feeds other hosts names of non-existent members. Possible starting points would involve building reputations of end-systems over time, and potentially adopting a more centralized design involving a trusted source.

7.2.3 Hybrid Architectures

This thesis has primarily focused on application end-point architectures that do not involve support from infrastructure. A natural direction involves exploring hybrid architectures that both utilize resources at application end-points as well as infrastructure nodes. A particularly interesting design point that we discussed in Chapter 4 involves waypoint architectures. Waypoints refer to machines that may be employed in addition to end systems actually participating in our broadcast. The waypoints are not statically provisioned nodes, however the system can invoke them dynamically, and leverage them to improve performance. Waypoints can improve the performance of the system by increasing the resources available, providing more stable nodes, and by providing nodes that can provide better performance quality. Several interesting questions arise with waypoints: when should a system dynamically invoke waypoints? should waypoints be invoked at particular geographical locations? Designing a waypoint service that can be shared among a large number of applications raises very interesting issues. How should the resources of waypoints be distributed among competing applications? What kind of charging model could be used to charge a particular application instantiation for the use of a set of resources?

7.2.4 Protocol Design Enhancements

The design of protocols for End System Multicast is still an evolving field. There are a large number of dimensions which protocol designs must further consider. As one example, we have shown the importance of considering heterogeneity in node capabilities as part of protocol design through techniques for differential treatment. Further issues involve considering heterogeneity in other dimensions such as the duration nodes stay and the performance the nodes provide as parents. Interesting issues arise with having to simultaneously consider a range of metrics that may be in conflict with each other. On the other hand, there are several issues that need to be tackled with the design of very large scale groups. These include group management algorithms for tackling flash crowds, algorithms for clustering nodes based on geographical location by relying on techniques such as GNP [71], and repair mechanisms to improve quality for data delivery.

7.2.5 Understanding Trade-offs between Protocol Design Choices

While there has been a plethora of designs for overlay multicast, and overlay networks in general, there is relatively little insight on how protocols compare, and how various design alternatives compare. Several questions of this nature remain yet to be answered. What are the trade-offs between performance-centric and DHT-based protocols? What does it mean for a protocol design to be scalable? How to characterize the benefits of an approach that employs redundancy compared to one does not? What are the limitations of multi-tree data delivery approaches [43, 8]? How do they compare to alternate mesh-based approaches such as [14]? The answers to these questions are linked to simulation and emulation methodologies that can help scientifically analyze and compare overlay designs. An associated challenge is developing simulation and emulation models at an appropriate level of realism - while current simulators are often too far removed from reality and can potentially mislead, it is next to impossible to model all the complexity inherent in the Internet. One direction is to see how best to make use of the traces and data obtained as part of our deployment experience in running trace-driven simulation experiments that can enhance the realism of current simulators.

7.2.6 System Issues

There are several research questions that have originated from our effort to build a full-fledged broadcasting system. One question relates to the transport protocol in the system. It is clear from our experience that there is need to use a UDP-based transport protocol, both from the perspective of supporting NATs, and given that we are dealing with streaming media applications However, we believe traditional TCP-friendly UDP-based congestion control protocols may not be appropriate, and there is need to explore protocols that adapt at coarser time-scales. Another direction is to explore mechanisms such as buffering to improve the resilience of the system, given that we are dealing with broadcasting applications that are real-time but not interactive, and can tolerate higher latencies. A third direction is the design of techniques that can accurately determine the access bandwidth of hosts rather than relying on user configurations. Such a technique must also consider hosts in sites that are multi-homed and potentially have different access bandwidths along different Internet connections.

7.2.7 Management of Large-scale Distributed Systems

The deployment of a large-scale distributed and self-organizing broadcasting system raises important concerns regarding how such a system can be monitored. Accidental bugs in code can lead to serious implications given that we are dealing with high bandwidth traffic being exchanged between hosts without explicit control on who they are communicating with. Similar concerns apply in the context of other deployments of distributed systems, such as Planetlab [46]. Interesting research directions include the development of distributed monitoring infrastructures, that monitor a large range of invariants concerning the overall health of the system such as the total traffic going out of a node and high volumes of traffic being sent to a node from many different nodes.

Bibliography

[1] Enabling Live Internet Broadcasting using an Application Endpoint Architecture, Ph.D Thesis, Yang-hua Chu, Carnegie Mellon University.

[2] Akamai. http://www.akamai.com/.

[3] A.Legout and E.W.Biersack. Pathological behaviours for RLM and RLC. In *Proceedings of Nossdav '00*, 2000.

[4] Allcast. http://www.allcast.com/.

[5] Kevin C. Almeroth and Mostafa H. Ammar. Characterization of mbone session dynamics: Developing and applying a measurement tool. Technical Report GIT-CC-95-22, 1995.

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

[7] S. Casner and S. Deering. First IETF Internet Audiocast. *ACM Computer Communication Review*, pages 92–97, 1992.

[8] M. Castro, P. Druschel, A. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Content Distribution in Cooperative Environments. In *Proceedings of SOSP*, 2003.

[9] Chaincast. http://www.chaincast.com/.

[10] Y. Chawathe. Scattercast: An Architecture for Internet Broadcast Distribution as an Infrastructure Service. *ACM Multimedia Systems Journal on Multimedia Distribution*, 2002.

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

[12] Chu et. al. Early Deployment Experience with an Overlay Based Internet Broadcasting System. In *USENIX Annual Technical Conference*, June 2004.

[13] I. Clarke, O. Sandberg, B. Wiley, and T. W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *Proceedings of the ICSI Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[14] J. Albrecht D. Kostic, A. Rodriguez and A. Vahdat. Bullet: High Bandwidth Data Dissemination Using an Overlay Mesh. In *Proceedings of SOSP*, 2003.

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

[16] C. Diot, B.N. Levine, B. Lyles, H. Kassan, and D. Balsiefien. Deployment Issues for the IP Multicast Service and Architecture. In *IEEE Network, special issue on Multicasting*, 2000.

[17] K.Hua D.Tran and T.Do. "ZIGZAG: An Efficient Peer-to-Peer Scheme for Media Streaming". In *Infocom*, 2003.

[18] End System Multicast Toolkit and Portal. http://esm.cs.cmu.edu/.

[19] Rajagopla-Rao et al. "Load Balancing in Structured P2P Systems". In *Proceedings of the Second International Workshop on Peer-to-Peer Systems (IPTPS)*, 2003.

[20] C. Faloutsos, M. Faloutsos, and P. Faloutsos. On Power-Law Relationships of the Internet Topology. In *Proceedings of ACM Sigcomm*, August 1999.

[21] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Multicast Routing in Internetworks and Extended LANs. In *Proceedings of the ACM SIGCOMM*, August 2000.

[22] National Laboratory for Applied Network Research. Routing data. http://moat.nlanr.net/Routing/rawdata/.

[23] Cooperative Association for Internet Data Analysis. Mapnet project. http://www.caida.org/Tools/Mapnet/Data/.

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

[25] Gnutella. http://www.gnutella.com/.

[26] Groove networks. http://www.groovenetworks.com/.

[27] V. Hardman, A. Sasse, M. Handley, and A. Watson. Reliable Audio for Use over the Internet: A Packet Loss Robust-Audio Tool for Use over the Mbone. In *Proceedings of INET*, June 1995.

[28] D.A. Helder and S. Jamin. Banana Tree Protocol, an End-host Multicast Protocol, 2000. Technical Report.

[29] H.W.Holbrook and D.R. Cheriton. IP multicast channels: EXPRESS support for large-scale single-source applications. In *Proceedings of the ACM SIGCOMM 99*, August 1999.

[30] Inktomi. http://www.inktomi.com/.

[31] I.Stoica, D.Adkins, S.Zhuang, S.Shenker, and S.Surana. "Internet Indirection Infrastructure". *"IEEE/ACM Transactions on Networking"*, April 2004.

[32] V. Jacobson and S. McCanne. Visual Audio Tool (vat). In *Audio Tool (vat), Lawrence Berkley Laboratory.* Software on-line, ftp://ftp.ee.lbl.gov/conferencing/vat.

[33] S. Jain, R.Mahajan, D.Wetherall, G.Borriello, and S.D. Gribble. Scalable Self-Organizing Overlays. Technical report UW-CSE 02-02-02, University of Washington, February 2002.

[34] 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.

[35] G. Kortsarz and D. Peleg. Generating Low-Degree 2-Spanners. *SIAM Journal on Computing, Volume 27, Number 5.*

[36] J. Liebeherr and M. Nahas. Application-layer Multicast with Delaunay Triangulations. In *IEEE Globecom*, November 2001.

[37] A.M. Kermarrec M. Castro, P. Druschel and A. Rowstron. Scribe: A Large-Scale and Decentralized Application-Level Multicast Infrastructure. In *IEEE Journal on Selected Areas in Communications Vol. 20 No. 8*, Oct 2002.

[38] S. McCanne, E. Brewer, R. Katz, L. Rowe, E. Amir, Y. Chawathe, A. Coopersmith, K. Mayer-Patel, S. Raman, A. Schuett, D. Simpson, A. Swan, T. L. Tung, D. Wu, and B Smith. Toward a Common Infrastucture for Multimedia-Networking Middleware. In *Proceedings of NOSSDAV*, 1997.

[39] S. McCanne and V. Jacobson. Vic: A Flexible Framework for Packet Video. In *ACM Multimedia*, November 1995.

[40] M.S.Kim, S.S.Lam, and D.Lee. "Optimal Distribution Tree for Internet Streaming Media,". In *Proceedings of IEEE ICDCS*, May 2003.

[41] Napster. http://www.napster.com/.

[42] V. N. Padmanabhan, H. J. Wang, and P. A. Chou. "Supporting Heterogeneity and Congestion Control in Peer-to-Peer Multicast Streaming". In *Proceedings of the Third International Workshop on Peer-to-Peer Systems (IPTPS)*, 2004.

[43] V.N. Padmanabhan, H.J. Wang, and P.A Chou. Resilient Peer-to-peer Streaming. In *Proceedings of IEEE ICNP*, November 2003.

[44] Peercast. http://www.peercast.org/.

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

[46] Planetlab. http://www.planet-lab.org/.

[47] Quicktime. http://www.apple.com/quicktime.

[48] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *Proc. of ACM SIGCOMM*, 2001.

[49] Sylvia Ratnasamy, Mark Handley, Richard Karp, and Scott Shenker. Application-level Multicast using Content-Addressable Networks. In *Proceedings of NGC*, 2001.

[50] Real Broadcast Network. http://www.real.com/.

[51] Y. Rekhter and T. Li. A Border Gateway Protocol 4 (BGP-4), RFC 1771, March 1995.

[52] R. Renesse, Y. Minsky, and M. Hayden. A Gossip-Style Failure Detection Service. Technical Report TR98-1687, Cornell University Computer Science, 1998.

[53] R.Gopalakrishnan, J. Griffioen, G. Hjlmtsson, and C.J. Sreenan. Stability and Fairness Issues in Layered Multicast. In *Proceedings of Nossdav '99*, 1999.

[54] R.Gopalakrishnan, J. Griffioen, G. Hjlmtsson, C.J. Sreenan, and S. Wen. A Simple Loss Differentiation Approach to Layered Multicast. In *Proceedings of Nossdav '99*, 1999.

[55] A. Rowstron and P. Druschel. Pastry: Scalable, Distributed Object Location and Routing for Large-Scale Peer-to-Peer Systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[56] Rowstron, A., Castro, M., et. al. SimPastry (Scribe) Implementation, v3.0a.

[57] B. Bhattacharjee S. Banerjee and C. Kommareddy. Scalable Application Layer Multicast. In *Proceedings of ACM SIGCOMM*, August 2002.

[58] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A measurement study of peer-to-peer file sharing systems. In *Proceedings of Multimedia Computing and Networking (MMCN)*, January 2002.

[59] 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.

[60] S.Banerjee, S.Lee, B.Bhattacharjee, and A.Srinivasan. "Resilient Multicast using Overlays,". In *ACM Sigmetrics*, 2003.

[61] F.B. Schneider. Byzantine Generals in Action: Implementing Fail-stop Processors. *ACM transactions on Computer Systems, 2(2)*, pages 145–154, 1984.

[62] S.G.Rao, A.Bharambe, V.Padmanabhan, S.Seshan, and H.Zhang. "The Impact of Heterogeneous Bandwidth Constraints on Protocols for Overlay Multicast, Technical Report".

[63] Slashdot. http://www.slashdot.org/.

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

[65] Sorenson. http://www.sorenson.com/.

[66] S.Q.Zhuang, B.Y.Zhao, J.D.Kubiatowicz, and A.D.Joseph. Bayeux: An Architecture for Scalable and Fault-tolerant Wide-area Data Dissemination. In *Proceedings of NOSSDAV*, Apr 2001.

[67] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications. In *Proc. of ACM SIGCOMM*, 2001.

[68] I. Stoica, T.S.E. Ng, and H. Zhang. REUNITE: A Recursive Unicast Approach to Multicast. In *Proceedings of IEEE INFOCOM'00*, Tel-Aviv, Israel, March 2000.

[69] Streamer. http://streamerp2p.com/.

[70] 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.

[71] T.S.E.Ng and H.Zhang. "Predicting Internet Network Distance with Coordinates-Based Approaches". In *Infocom*, 2002.

[72] T.S.E.Ng, Y.Chu, S.G. Rao, K.Sripanidkulchai, and H.Zhang. "Measurement-Based Optimization Techniques for Bandwidth-Demanding Peer-to-Peer Systems". In *Infocom*, 2003.

[73] W. Wang, D. Helder, S. Jamin, and L. Zhang. Overlay Optimizations for End-host Multicast. In *Proceedings of Fourth International Workshop on Networked Group Communication (NGC)*, October 2002.

[74] Z. Wang and J. Crowcroft. Bandwidth-Delay Based Routing Algorithms. In *IEEE GlobeCom*, November 1995.

[75] E. W. Zegura, K. L. Calvert, and S. Bhattacharjee. How to Model an Internetwork. In *Proceedings of IEEE Infocom*, March 1996.

[76] B. Zhang, S. Jamin, and L. Zhang. Host Multicast: A Framework for Delivering Multicast to End Users. In *Proceedings of IEEE Infocom*, June 2002.