Enabling Live Internet Broadcasting Using an Application Endpoint Architecture

Yang-hua Chu

CMU-CS-05-133
May 9, 2005

School of Computer Science
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Hui Zhang, Chair
Srinivasan Seshan
Peter Steenkiste
Jörg Liebeherr, University of Virginia

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

This research was sponsored in part by the National Science Foundation under grant nos. ANI-0326472, ANI-0085920, ANI-0331653, the US Air Force Research Laboratory under grant no. F306029910518, and a generous fellowship from the Intel Corporation. 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 any sponsoring institution, the U.S. government or any other entity.

Keywords: Internet Architecture, Overlay Networks, Peer-to-peer Systems, Application-level Multicast, Video Broadcasting, Multimedia Streaming, Economic Incentive, Taxation.

Abstract

It has been a long standing challenge to make Internet audio/video broadcasting a *commodity service*. This means that anyone with commodity Internet connection and computer equipments can broadcast high quality video to a large group of receivers in real time. The key challenge is the bandwidth cost in distributing the video streams. To distribute 300Kbps video stream to 100 receivers directly, a publisher must provision 30Mbps bandwidth to the Internet. This is too expensive for most individuals to afford.

The conventional wisdom is to add functionality in the underlying network infrastructure, i.e. at the IP layer. With IP Multicast, the publisher sends just one copy of the video stream to the IP network, and the network intelligently replicates the video streams to all the receivers. By shifting the task of data replication to the IP routers, IP Multicast greatly reduces the bandwidth requirements for the publishers and receivers. However, 15 years after its initial proposal, IP Multicast is still plagued with concerns pertaining to scalability, network management, deployment, and support for higher level functionality.

This dissertation takes a different architecture approach to meet the challenge in broadcasting high quality video over the Internet. Our thesis is that "it is feasible *today* to provide video broadcasting as a *commodity service*, *without changing the underlying IP infrastructure*." We propose a new architecture called End System Multicast. In End System Multicast, data replication is performed not by the routers, but by the receivers in the broadcast, which are end systems on the Internet. Thus, the publisher only needs to send the video stream to a few receivers, and these receivers iteratively forward the video streams to other receivers. This avoids costly bandwidth provisioning for the publisher and requires no changes to the IP infrastructure.

We demonstrate the feasibility of End System Multicast not only in simulators and Internet testbeds, but also in live broadcast scenarios. In the past two years, we have built an operational video broadcasting system based on this architecture. The system has been successful in broadcasting 20 events, benefiting 4000 users.

This dissertation describes a complete solution in building a video broadcast system based on End System Multicast. This includes protocol-level designs to handle group dynamics and network heterogeneity, and system-level designs to integrate with media codecs and players. In addition, we address the issue of incentive for publishers and receivers to participate in this system.

To my late mother, Da-hon Fu Chu

Acknowledgements

First and foremost, I thank my advisor, Prof. Hui Zhang. Throughput the years he offered valuable guidance both in research and in personal development. He has deep understanding about the Internet architecture and its problems. This dissertation work started in 1998 when he questioned the conventional wisdom that multicast was implemented at the IP layer. Moreover, he has high expectation to drill a research topic in great depth. We spent the next six years convincing ourselves (and others) that an end system approach to multicast is a better architecture. In the process he taught me how to be a researcher. As an advisor he took interests in my personal growth. He considered my circumstance and offered advice that excelled me not only in research but also in personal development. I am grateful to have him as an advisor.

I thank other members of my thesis committee, Prof. Jörg Liberherr, Prof. Srini Seshan, and Prof. Peter Steenkiste, for their valuable feedbacks. Jörg flew from U. Virginia to attend both the thesis proposal and thesis defense. I appreciate his critical and unbiased inputs on both the system architecture and protocol semantics. Peter and Srini have graciously sit through many *ESM* talks at CMU over the years and offered constructive feedbacks. Special thanks to Prof. John Chuang for his guidance on the incentive part of the dissertation work. His breadth of knowledge in networking and economics is an eye opener.

A majority of this dissertation is joint work with a fellow Ph.D. student Sanjay Rao. We collaborated on *ESM* since 1998. We spent many wonderful hours brainstorming ideas and shredding the ideas in pieces. He is a research partner anyone could hope for. I admire and learn from his dedication in research and his rigor in the thinking process.

It was a privilege to work closely with other Ph.D. students and staff members in the *ESM* project: Aditya Ganjam, Eugene Ng, Kunwadee Sripanidkulchai, and Jibin Zhan. Much of the work in Chapter 3 is joint work with them. We spent several months putting together the broadcast system, and several more months to analyze the event trace. We shared the excitement of making the system real as well as the burden of maintaining the system. I learned a lot about working together as a team; we can achieve so much greater collectively as a whole than separately as individuals.

I am honored to work with many talented Master students and undergraduates over the years: Frank Chan, Annie Cheng, Brian Goodman, Tian Lin, Jiin Joo Ong, Chris Palow, Vishal Soni, and Sean Wang. They greatly contribute to the working of the ESM system at different stages. Moreover, each of them brings an unique perspective toward the research work. I learned from them as much as they learned from me.

Pittsburgh is great city to live in but the people I have been with makes the times in Pittsburgh even more worthwhile. Thanks to Jun Gao, Qifa Ke, An-Cheng Huang, Andy Myers, Rob O'Callahan, Ion Stoica, Yinglian Xie, Shuheng Zhou, and many others for the time we spent together. There are good memories of great food, fun conversations, and healthy exercises. Special thanks to Rob who has given me valuable advice throughput the years.

I am blessed to have a loving family. My mom passed away in 1998 early in my Ph.D career. This dissertation is dedicated to her. I think she is very happy that I finish the degree. My dad is in many ways my role model. He strongly believes in education. He is himself a Ph.D. and I am happy to follow his footstep. I thank my brothers, Yung-hua, Paul, and Hao, and my sister-in-laws, Julia, Suzanne, and Emily for their unconditional support. Even though we live far apart, they are always there for me when I need it.

I thank our daughter Ella, who was born in November 5, 2004. She gave me the final momentum to put together the dissertation and defend on October 20, 2004. I look forward to sharing our lives together.

Finally, I thank my wife Wendy. We share each other's joy and live through the sadness. Even though she still could not understand why a "doctor" degree takes 7 years to finish (she earned a "doctor" of Judiciary in 3 years), she gave me unconditional support to complete this dissertation. I am glad that I don't have to answer this question after now :)

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Before the Internet, information publishing used to be a *privilege* rather than a *commodity*. The cost of printing and distribution is so high that only a few people can afford. Internet *commoditizes* information publishing; it brings the cost of publishing to almost zero. Through the World Wide Web, individuals are empowered to publish information that can be shared by potentially millions of people on the Internet, with almost no cost.

Unfortunately, the Internet today is handicapped to publish only text and images, but not live video. The key challenge is bandwidth. A 10-minute 300Kbps video clip contains 22MB of data, which is orders of magnitudes more than typical text and images. The conventional server-based solution for the Web, as shown in Figure 1.1(a), is no longer applicable. For instance, to broadcasts 300Kbps video streams to 100 receivers simultaneously, a publisher must provision 30Mbps bandwidth to the Internet. This is still far too expensive for most individuals to afford.

The goal of this dissertation is to enable live Internet broadcasting as a *commodity* service. By commodity, I mean anyone can broadcast their live video online, as long as the person has a "high-speed" Internet connection, a camera, and a computer. Any number of receivers can view the broadcast stream, regardless of their network speed and connectivity constraints.

Providing this service has been a long standing challenge in the network research over the past decade. In his seminal work in 1989 [22], Deering proposes to add functionality in the underlying network infrastructure. With IP Multicast, the publisher sends just one copy of the video stream to the IP network, and the network intelligently replicates the video streams to all the receivers. This is shown in Figure 1.1(b). By shifting the task of data replication to the routers, IP Multicast greatly reduces the bandwidth requirements for the publishers and receivers. However, 15 years after its initial proposal, IP Multicast is still plagued with concerns pertaining to scalability (to the number of groups), network management, deployment, and support for higher level functionality.

In summary, the key challenge for video broadcasting is replicating high bandwidth video streams to a large number of receivers. In the conventional server-based solution, the video stream is replicated at the server, which requires costly bandwidth provisioning. IP Multicast advocates to replicate video streams inside the network. However, it requires changes to the underlying networking infrastructure, which is difficult from both technical and logistical perspectives. *The question is, can Internet provide video broadcasting as a commodity service, without costly provisioning at the server* **and** *without changing the underlying network infrastructure?*

In this dissertation, we answer this question affirmatively. To provide such a service, we propose a new architecture called *End System Multicast*. In this architecture, data replication is performed

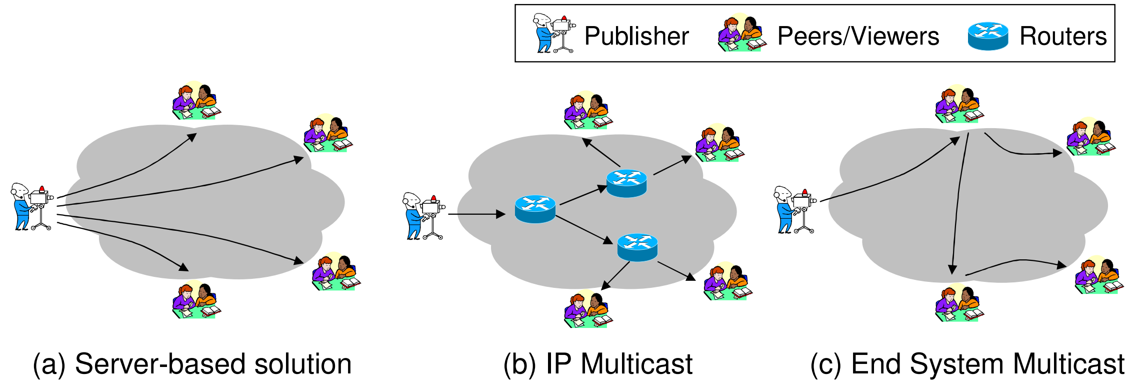

Figure 1.1: Examples to illustrate three different architecture solutions for Internet broadcasting. Our dissertation advocates for End System Multicast as shown in (c).

not by the routers, but by the receivers in the broadcast, which are end systems on the Internet. Thus, the publisher only needs to send the video stream to a few receivers, and these receivers iteratively forward the video streams to other receivers, as shown in Figure 1.1(c). This greatly reduces the bandwidth requirement for the publisher. Moreover, since end systems communicate using only unicast IP service, there is no change to the IP infrastructure.

1.1 Main Contribution

The main contribution of this dissertation is to provide the first complete solution that offers video broadcasting as a commodity service on the Internet, without making any changes to the existing network infrastructure. The dissertation is composed of three main components:

End System Multicast Architecture: We challenge the conventional wisdom that IP Multicast is the architecture for wide-area Internet multicast/broadcast. We propose a new architecture called End System Multicast, where end systems, and not routers, perform data replication. However, the key concern with the architecture is the performance penalty. In particular, End System Multicast introduces duplicated packets and incurs transient data loss when adapting to network congestion and membership changes. To study the performance implication of this architecture, we have designed one of the first self-organizing protocols (called Narada). Our evaluation results from simulation and Internet testbed experiments indicate that the performance penalties are low from both the application and the network perspective.

Broadcast System Design, Implementation, and Deployment: To demonstrate that End System Multicast is a feasible architecture for video broadcast, we design, implement, and deploy a broadcast system based on this architecture. To our knowledge this is among the first system with real application deployment and experience based on End System Multicast. The system has been in operation for over two year, and it is used by over 4000 users in 20 events. The post-mortem analysis of the event traces was positive, and provides valuable insight for future improvements. We believe our experience offers a good roadmap for others to design and deploy future Internet broadcast systems.

Incentive Mechanisms: For the End System Multicast architecture to succeed, the premise is that the viewers are willing to contribute bandwidth resource and participate in data replication. While in many cases it is reasonable to assume receivers are altruistic in contributing resource, we consider the scenario in which all receivers are strategic agents, i.e. they contribute more resources only if they see clear benefits in doing so. We propose a new *taxation* scheme, where resource-rich peers subsidizes for the resource-poor receivers. Our simulation results indicate that taxation can achieve good social welfare without incurring a significant overhead to the system.

1.2 Background

In this section, we provide background information about the demand and characteristics of video broadcast. We then review the IP Multicast approach, and point out the fundamental concerns of the approach.

1.2.1 Video Broadcast: Demand and Characteristics

Consider a scenario where a university holds a series of prestigious lectures, with talks given by eminent scientists, professors and industry leaders. As part of its outreach program, the university wants to broadcast these lectures live on the Internet. The expected audience include alumni of the university scattered all over the world, members of the local industry, high schools and colleges around the world who normally may not have access to such prestigious presenters. These broadcast scenarios typically share the following characteristics:

- *Single source:* A typical broadcast has a single source (speaker), talking for most of the broadcast. The live broadcast could further be enhanced with Internet chat tools where participants can discuss the lecture, and perhaps could even raise questions that could be forwarded to a speaker using a moderator.

- *Dynamic group membership:* A typical broadcast conference or lecture lasts from tens of minutes to several hours. Although we expect individual participants to join and leave at various points during the broadcast, receivers typically participate at least for a few minutes as they are interested in the content of the broadcast.

- *Medium to large scale:* A typical broadcast may involve hundreds and possibly thousands of receivers. Moreover, our experience indicates that the group size may fluctuate within one broadcast event. If a event has multiple speakers, the participation may fluctuate from one speaker to another.

- *High bandwidth:* A high quality, real-time video streaming requires high bandwidth. Our experience indicates that a reasonable quality motion audio/video stream takes at least 300Kbps to encode. However, some hosts cannot sustain this video bitrate due to transient network congestion or limited link speed. In this case, video streams must be degraded to accommodate a lower bitrate.

1.2.2 IP Multicast

IP Multicast, first proposed in 1988 [22], was motivated by the use of multicast in the local area network (LAN) environment. In the LAN environment, multicast makes distributed applications more *efficient* and *robust*. It is more efficient because multicast reduces the transmission overhead on the sender and the network. It is more robust because distributed applications can locate and share information with one another without knowing their addresses ahead of time.

Supporting multicast is easier in a LAN environment than on the wide area network (WAN). Typically, hosts on the same LAN share a common transmission channel, such as Ethernet. Thus, the mechanism to deliver unicast data is the same as multicast data and so is the cost to the network. However, in a store-and-forward IP architecture, supporting an efficient and robust multicast service is not a trivial task.

In IP Multicast, routers in the network participate in the IP Multicast protocol and build multicast trees between the sources and their respective receivers. Multicast data sent by the source host is replicated along the multicast tree by the routers at the splitting points. IP Multicast is highly efficient: only a single packet traverses each physical link at most once. Moreover, the delay from the source to each receiver is as short as its unicast delay.

However, despite the enormous efforts from both the research and industry communities, IP Multicast has yet been widely deployed on the Internet due to several fundamental concerns:

- *Scaling concerns with per-group state*: IP Multicast requires routers to maintain per group state, which introduces high complexity and serious scaling concerns at the IP layer.

- *Vulnerable to flooding attack*: The current IP Multicast model allows for an arbitrary source to send data to an arbitrary group. This makes the network vulnerable to flooding attacks by malicious sources, and complicates network management and provisioning.

- *Limited address space*: IP Multicast requires every group to dynamically obtain a globally unique address from the multicast address space and it is difficult to ensure this in a scalable, distributed and consistent fashion.

- *Difficult to support higher-level functionality*: IP Multicast is a best effort service. Providing higher level features such as reliability, congestion control, flow control, and security has been shown to be more difficult than in the unicast case.

- *High deployment barrier*: IP Multicast calls for changes at the infrastructure level, and this slows down the pace of deployment.

While there have been attempts to partially address some of the issues at the IP layer [33, 57, 74], fundamental concerns pertaining to the “stateful” architecture of IP Multicast and support for higher layer functionality have remained unresolved.

1.3 Organization

The rest of the dissertation is organized as follows. Chapter 2 describes the End System Multicast architecture and a proof-of-concept Narada protocol. We argue the viability of the End System

Multicast architecture by comparing the performance tradeoff between Narada and IP Multicast.

Chapter 3 describes the broadcast system based on End System Multicast, and our experience deploying the system in real broadcast events. The chapter addresses the system-level issues of integrating an generic overlay multicast protocol in a real application, including receiver heterogeneity and network congestion, NAT/firewall, front-end interface to publishers/viewers, and testing/logging. The system has been used to broadcast lectures and conferences, and the chapter also describes the analysis methodology, analysis results, and the lessons learned.

Chapter 4 addresses the issue of incentive for publishers and receivers to participate in the broadcast system. We consider the scenario in which all receivers are strategic agents, and propose a new taxation scheme, where resource-rich peers subsidizes for the resource-poor receivers.

Finally, Chapter 5 summarizes the dissertation, discuss its limitation, and point directions for future work.

1.4 Joint Work and Acknowledgement

Research in End System Multicast spans over several years. Many people are involved in this project, and some continue to pursue this research as I am wrapping up the dissertation work. When writing this dissertation I find it difficult to separate my own work from others. So in this dissertation, I present all the research results where I have taken significant leadership both intellectually and in the actual execution of the research work. Below I wish to explicitly acknowledge and highlight aspects of joint work.

There are three main chapters in this dissertation. Chapter 2 is joint work with fellow student Sanjay Rao. Chapter 3 is joint work with Aditya Ganjam, Eugene Ng, Sanjay Rao, Kunwadee Sripanidkulchai, Jibin Zhan. In addition, several Master students and undergraduates contribute significantly to the design and implementation of the broadcast system (see Acknowledgement). Chapter 4 is my work alone but with significant contributions from Prof. John Chuang with respect to problem formulation. All of the work in this dissertation is supervised by my advisor Prof. Hui Zhang.

Chapter 2

End System Multicast

In response to the serious scalability and deployment concerns with IP Multicast, we propose a new architecture called End System Multicast [17]. In this architecture, all multicast functionality is pushed to the end systems. This shifting of multicast support from the routers to end systems has the potential to address most problems with IP Multicast. However, the key concern is the performance penalty associated with such an architecture. In particular, End System Multicast introduces duplicated packets and incurs larger end-to-end delay than IP Multicast.

In this chapter, we study these performance concerns in the context of our a proof-of-concept protocol called Narada. Narada is a self-improving protocol that optimizes for delay and bandwidth in a fully distributed fashion. We evaluate the performance of Narada in simulation an Internet testbeds. Our results indicate that the performance penalties are low both from the application and the network perspectives.

To contrast with IP Multicast, this chapter intentionally adopts a *generic* definition of end systems. In practice, there is a wide spectrum of end systems on the Internet. They range from well-provisioned infrastructure service nodes such as Akamai proxies [2], to commodity user machines behind mainstream cable/DSL connections. At the end of this chapter, we argue that the choice of end systems can have non-trivial implication on the protocol design, performance, and deployment path. We discuss different architecture instantiation of the end systems, and justify our choice in the context of the video broadcast applications.

The rest of the chapter is organized as follows. We begin by describing the architecture in Section 2.1, and compare the potential benefits and drawbacks of this architecture. Next in Section 2.2, we describe Narada, a proof-of-concept protocol for the architecture. We evaluate Narada in simulation and on Internet testbeds. The evaluation methodology and the experiment results are presented in Section 2.3. Finally, we discuss different choices of end systems and justify our choice in Section 2.6.

2.1 Architecture Overview

In End System Multicast, all multicast related functionality, including membership management, multicast routing, and packet duplication, are implemented at end systems, assuming only unicast IP service. End systems organize themselves into an overlay spanning tree for data delivery. The tree is an overlay in the sense that each edge in the overlay corresponds to an unicast path between

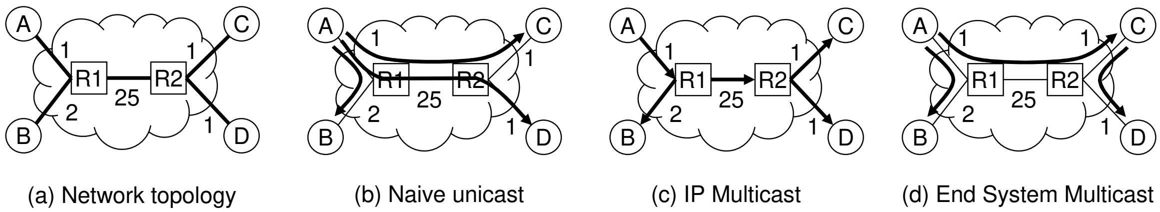

Figure 2.1: Example to illustrate naive unicast, IP Multicast and End System Multicast.

two end systems in the underlying Internet.

We illustrate the differences between IP Multicast, naive unicast and End System Multicast using Figure 2.1. Figure 2.1(a) depicts an example physical topology. $R1$ and $R2$ are routers, and A, B, C and D are end systems. The numbers indicate the link delays. $R1 - R2$ represents a costly transcontinental link. All other links are cheaper local links. Further, let us assume A is the source and wishes to send data to all other nodes.

Figure 2.1(b) depicts naive unicast transmission. Naive unicast incurs high data redundancy on links near the source. For example, link $A - R1$ carries three copies of a transmission by A. Moreover, naive unicast results in duplicate copies on costly links. For example, link $R1 - R2$ has two copies of a transmission by A.

Figure 2.1(c) depicts the IP Multicast tree constructed by DVMRP [22]. DVMRP is the classical IP Multicast protocol, where data is delivered from the source to recipients using an IP Multicast tree composed of the unicast paths from each recipient to the source. Redundant transmission is avoided, and exactly one copy of the packet traverses any given physical link. Each recipient receives data with the same delay as though A were sending to it directly by unicast.

Figure 2.1(d) depicts an "intelligent" overlay tree that may be constructed 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 over naive unicast based schemes has been obtained with absolutely no change to routers, and all intelligence is implemented at the end systems. However, while intelligently constructed overlay trees can result in much better performance than naive unicast solutions, they fundamentally cannot perform as well as solutions with native IP Multicast support. For example, in Figure 2.1(d), link $A - R1$ carries a redundant copy of data transmission, while the delay from source A to receiver D has increased from 27 to 29.

In summary, End System Multicast provides several advantages over IP Multicast:

- *Scalable to the number of groups*: End System Multicast uses unicast to replicate multicast data, and thus IP routers no longer need to maintain per-group state. Moreover, End System Multicast does not need any globally unique address at the IP layer, and hence avoids class D address assignment issues with IP Multicast.

- *Simplify support for higher layer functionality*: The splitting points are intelligent end systems with large storage and computation power. This allows application-specific customization at the replication points.

- *Immediately deployable*: To start a multicast group, the participating end systems just need to install compatible software. This is in contrast with IP Multicast. To deploy an ubiquitous multicast service, all routers on the Internet must be modified to support multicast at the IP layer.

However, there are two major concerns with End System Multicast from the architecture perspective:

- *Protocol complexity*: End systems have limited topological and performance information about the network. Thus, end systems must self-organize, discover, and exchange information among themselves to construct good overlays. Moreover, end systems are prone to host failure and network congestion. All these factors make the distributed protocol more challenging to build and scale to large group size.

- *Performance and Overhead*: An overlay approach to multicast, however efficient, cannot completely avoid introducing duplicated packets to the network and extra delays to the application. It is impossible to completely prevent multiple overlay edges from traversing the same physical link and thus some redundant traffic on physical links is unavoidable. Further, communication between end systems involves traversing other end systems, potentially increasing latency.

We illustrate examples the benefits of End System Multicast in the context of video broadcast applications in Chapter 3. In this chapter, we focus on the concerns with respect to the protocol complexity and performance.

2.2 Narada Protocol

We study the feasibility and performance concerns of End System Multicast in the context of the Narada protocol. This protocol is designed with the following objectives in mind:

- *Self-organizing*: The construction of the end system overlay is fully distributed. It is robust to dynamic changes in group membership.

- *Overlay efficiency*: The tree constructed is efficient from both the network and the application perspectives. From the network perspective, the constructed overlay ensures that redundant transmission on physical links is kept minimal. From the application perspective, the overlay satisfies the bandwidth and latency requirements of the applications.

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

- *Adaptive to network dynamics*: The overlay adapts to long-term variations in Internet path characteristics, while being resilient to network noise and inaccuracies that is inherent in the measurement of these quantities.

There are several approaches to construct overlay spanning trees for data delivery. Section 2.2.1 discusses the *mesh-first* approach that Narada uses. We explain the distributed algorithms that Narada uses to construct and maintain the mesh in Section 2.2.2. We present mechanisms Narada uses to improve mesh quality in Section 2.2.3. 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, and we discuss this in Section 2.2.4.

2.2.1 Overlay Structure

There are several ways overlays can be constructed for data delivery. In the following we discuss the *tree-first* and *mesh-first* approaches, as both of them are used in this dissertation. Other approaches are discussed in the related work in Section 2.4.

In the tree-first approach, members construct the tree directly. The construction is simple: members explicitly select their parents from among the members they know. In the mesh-first approach, members construct trees in a two-step process. First it constructs a richer connected graph (a *mesh*). In the second step, it constructs spanning trees of the mesh, each tree rooted at the corresponding source using well known routing algorithms.

We note that there is no clear-cut win between the tree-first and mesh-first approaches, as the choice likely depends on the underlying assumptions about the application. Narada chooses the mesh-first approach for the following reasons. (i) Narada wishes to be robust against a single point of failure. A tree-first approach typically requires a consensus of a *root* among members, which is a single point of failure. (ii) Constructing per-source tree out of a sparse mesh is a well-known problem. We can leverage standard routing algorithms for construction of data delivery trees, such as DVMRP. However, in designing the video broadcast system in Chapter 3, we have gradually evolved the Narada protocol from a mesh-first approach to a tree-first approach. In video broadcast, the source node is intrinsically a single point of failure. By relying on this assumption, group management and overlay construction can be greatly simplified.

The remainder of this chapter focuses on the mesh-first approach. In this approach, trees for data delivery are constructed entirely from overlay links chosen in the mesh. Hence, it becomes important to construct a good mesh so that good quality trees may be produced. In particular, we attempt to ensure the following properties: (i) there exist paths along the mesh between any pair of members such that the qualities of the paths are comparable to the qualities of the unicast path between the members. By path qualities, we refer to the metrics of interest for the application, and can be delay, bandwidth, or both. and (ii) each member has a limited number of neighbors in the mesh. Limiting the number of neighbors in the mesh controls the overhead of running routing algorithms on the mesh.

2.2.2 Group Management

In this section, we present mechanisms Narada uses to keep the mesh connected, 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 robust-

```
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 ≥ s_kj i
        then ignores the entry pertaining to k
        else i updates its entry for k to <k, s_kj, t>
```

Figure 2.2: Actions taken by a member i on receiving a refresh message from member j.

ness, 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 2.2.2.3.

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 2.2.

We note that because each member must maintain state for all other members to avoid a single point of failure, Narada is not scalable to a large group size. In video broadcasting applications, the source is fundamentally a single point of failure. Thus in designing the system for video broadcast (Chapter 3), the Narada protocol is modified to maintain only a subset of group members in exchange for better scalability and efficiency.

Given that a distance vector routing algorithm is run on top of the mesh (Section 2.2.4), routing update messages exchanged between neighbors can include member sequence number information with minimum extra overhead.

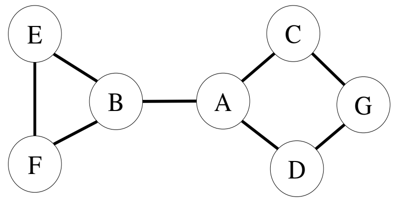

Figure 2.3: A sample virtual topology

2.2.2.1 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.

2.2.2.2 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 2.2.4, 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 [68], 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 2.3. 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 a refresh announcing stale information regarding a dead member. However, dead member information can be flushed after sufficient amount of time.

```
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 ≥ 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 2.4: Scheduling algorithm used by member i to repair mesh partition

2.2.2.3 Repairing Mesh Partitions

It is possible that member failure can cause the mesh to become partitioned. For example, in Figure 2.3, 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 carefully 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 2.4.

2.2.3 Improving mesh quality

The constructed mesh can be quite sub-optimal, 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

```
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 += (CL-NL)/CL
     end
  end
  return utility
```

Figure 2.5: Algorithm i uses in determining utility of adding link to j

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 2.5 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 expected gain in utility if a link to j is added as described in Figure 2.5. i decides to add a link to j if the expected utility gain exceeds a given threshold. The 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

```
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 2.6: Algorithm i uses to determine consensus cost to a neighbor j

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 2.6. 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 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.

2.2.4 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 [62]. 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 [22]. 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.

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 Evaluation

Our evaluation seeks to answer the following two questions:

- From the application perspective, what performance penalty does an End System Multicast architecture incur as compared to IP Multicast?
- From the network perspective, what is the overhead associated with End System Multicast in constructing good overlays?

To answer these question, we compare the performance of a scheme for disseminating data under the IP Multicast architectural framework, with the performance of various schemes for disseminating data under the End System Multicast framework. We describe these schemes in Section 2.3.1 and the performance metrics we use in evaluating the schemes in Section 2.3.2.

We have conducted our evaluation using both simulation and Internet experiments. Internet experiments help us understand how schemes for disseminating data behave in dynamic and unpredictable real-world environments, and give us an idea of the end-to-end performance seen by actual applications. On the other hand, simulations help analyze the scaling properties of the End System Multicast architecture with larger group sizes. Sections 2.3.3 and 2.3.4 present results for our Internet and simulation experiments.

2.3.1 Schemes Compared

We compare the following schemes for disseminating data in our simulation and Internet experiments.

- *DVMRP*: We assume that IP Multicast involves constructing classical DVMRP-like trees [22], composed of the reverse paths from the source to each receiver.
- *Narada*: This represents a scheme that construct overlay trees in an informed manner, making use of network metrics like bandwidth and latency. It is indicative of the performance one may expect with an End System Multicast architecture, though an alternate protocol may potentially result in better performance.
- *Random*: This represents a naive scheme that constructs random but connected End System Multicast overlays.
- *Naive Unicast*: Here, the source simultaneously unicasts data from the source to all receivers. Thus, in a group of size *n*, the source must send *n-1* duplicate copies of the same data, with *Naive Unicast*.

2.3.2 Performance Metrics

To facilitate our comparison, we use several metrics that capture both application and network level performance.

- *Latency*: This metric measures the end-to-end delay from the source to the receivers, as seen by the application. We also define *Relative Delay Penalty (RDP)*, which is a measure of the increase in delay that applications perceive while using Narada. RDP is a ratio of the delay between two members along the overlay to the unicast delay between them. Thus Naive Unicast has an RDP of 1, and DVMRP also has an RDP of 1, assuming symmetric routing.
- *Bandwidth* This metric measures the application level throughput at the receiver.
- *Stress* We refer to the number of identical copies of a packet carried by a physical link as the *stress of a physical link*. For example, in Figure 2.1, link $R1 - R2$ has a stress of 1 and link $A - R1$ has a stress of 2. In general, we would like to keep the stress on all links as low as possible.

  We also define the *Worst Case Stress* metric to measure the effectiveness of Narada in distributing network load across physical links. Worse case stress is 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. DVMRP has a worse case stress of 1. On the other hand, Naive Unicast has a worse case stress of r, where r is the number of receivers.

- *Resource Usage*: 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. Implicit here is the assumption that links with higher delay tend to be associated with higher cost. For example in Figure 2.1, the resource usage is 31 for the overlay tree and 29 for one possible DVMRP tree. We then compute the *Normalized Resource Usage (NRU)* of a scheme as the ratio of the resource usage with that scheme relative to the resource usage with DVMRP.

Latency and bandwidth are application level performance metrics, while all other metrics measure network costs. Not all applications care about both latency and bandwidth. Our evaluation thus considers the needs of applications with more stringent requirements (such as video broadcasting), which require both high bandwidth and low latencies. An architecture that can support such applications well can potentially also support applications that care about latency, or bandwidth alone.

2.3.3 Internet Experiments

Our Internet experiments are conducted on a wide-area test-bed of 13 hosts located at university sites in the U.S. and Canada. The varying nature of Internet performance influences the relative results of experiments done at different times. Ideally, we should test all schemes for disseminating data concurrently, so that they may observe the exact same network conditions. However, this is not possible, as the simultaneously operating schemes would interfere with each other. Therefore, we adopt the following strategy: (i) we interleave experiments with the various protocol schemes that we compare to eliminate biases due to changes that occur at shorter time scales, and (ii) we run the same experiment at different times of the day to eliminate biases due to changes that occur at a longer time scale. We aggregate the results obtained from several runs that have been conducted over a two week period.

Every individual experiment is conducted in the following fashion. Initially, all members join the group at approximately the same time. The source multicasts data at a constant rate and after four minutes, bandwidth and round-trip time measurements are collected. Each experiment lasts for 20 minutes. We adopt the above set-up for all schemes, except *Sequential Unicast*. In *Sequential Unicast*, we unicast data from the source to each receiver for two minutes in sequence. The source host, located at UCSB, is sending at a CBR traffic of 1.2 Mbps.

We begin by presenting our experimental methodology. We then present results in a typical experiment run in Section 2.3.3.1. Section 2.3.3.2 compare various schemes for constructing overlays with regard to application level performance and network cost. We also studied the performance of Narada under different source rates and host sets. These results are presented in [18].

2.3.3.1 Results with a Typical Run

The results in this section give us an idea of the dynamic nature of overlay construction, and how the quality of the overlay varies with time.

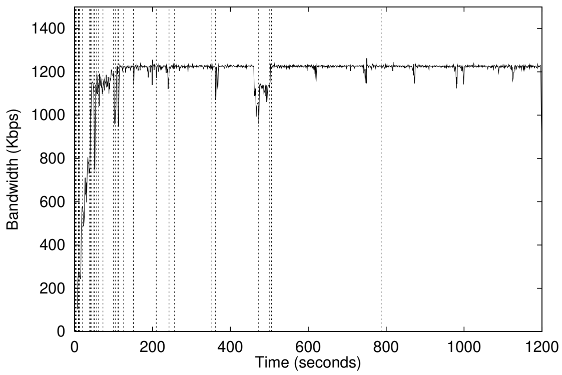

Figure 2.7: Mean Bandwidth averaged over all receivers as a function of time.

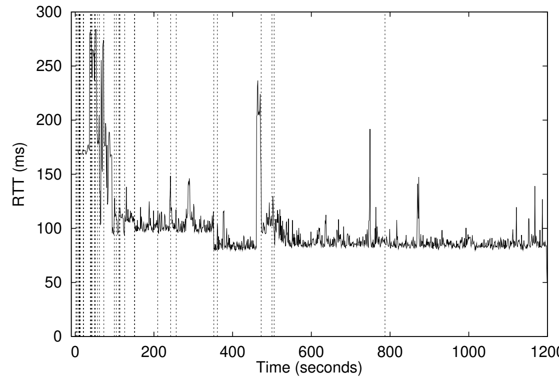

Figure 2.8: Mean RTT averaged over all receivers as a function of time.

Figure 2.7 plots the mean bandwidth seen by a receiver, averaged across all receivers, as a function of time. Each vertical line denotes a change in the overlay tree for the source UCSB. We observe that it takes about 150 seconds for the overlay to improve, and for the hosts to start receiving good bandwidth. After about 150 seconds, and for most of the session from this time on, the mean bandwidth observed by a receiver is practically the source rate. This indicates that all receivers get nearly the full source rate throughout the session.

Figure 2.8 plots the mean RTT to a receiver, averaged across all receivers as a function of time. The mean RTT is about 100 ms after about 150 seconds, and remains lower than this value almost throughout the session. We have also evaluated how the RTTs to individual receivers vary during a session and results are presented in [18]. 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.

Figures 2.7 and 2.8 also highlight two key performance concerns with Narada:

- *Long startup time*: When members first join the group, they do not know any network information. Thus the initial overlay is likely a random mesh with suboptimal performance in bandwidth and RTT.

- *Performance degradation due to network conditions*: 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.

These performance concerns have since been addressed by us and other researchers. To improve long startup time, we have investigated probing heuristics to select good neighbors quickly [52]. Our finding is that light-weight RTT probing is effective in predicting bandwidth. Thus by incorporating RTT probing, the protocol can shorten the startup time to less than 20 seconds for 90% of the joining members with similar group size.

There are several research works that aim to reduce the impact of network congestion and group dynamics on application performance. For example, [53, 11] use multiple disjoined path increase

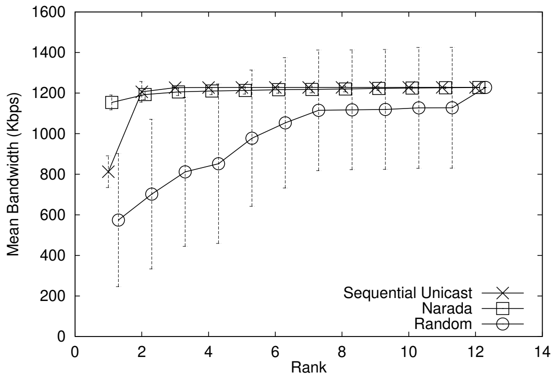

Figure 2.9: Mean bandwidth versus rank at 1.2 Mbps source rate

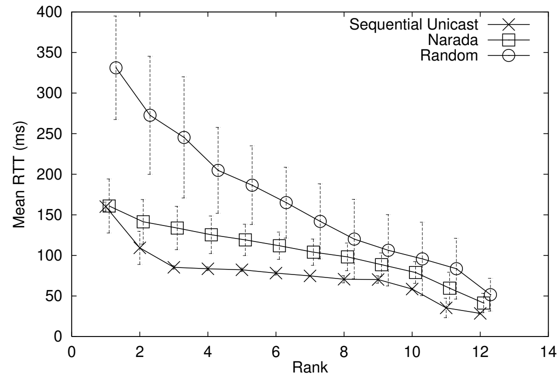

Figure 2.10: Mean RTT versus rank at 1.2 Mbps source rate

data resiliency. [5] actively buffers and repairs data packets from other members when the overlay structure is in transient. In Chapter 3, we use multiple video bitrates and prioritized forwarding to cope with transient performance degradation.

Given the startup time we observe here, we gather bandwidth and RTT statistics after four minutes for the rest of our experiments.

2.3.3.2 Comparision with Other Schemes

We present the results that compare the bandwidth and latency performance of various schemes for disseminating data on the Internet. In evaluating the performance of various schemes, we are not just interested in how each receiver perform for individual runs; we are more interested in how an overlay tree perform as a whole. Therefore, we introduce the notion of a *rank* that captures the performance of an 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.

Figure 2.9 plots the mean bandwidth against rank for three different schemes. Each curve corresponds to one scheme, and each point in the curve corresponds to the mean bandwidth that a machine of that rank receives with a particular scheme, averaged across all runs. The error-bars show the standard deviation. Thus they do not indicate confidence in the mean, rather they imply the degree of variability in performance that a particular scheme for constructing overlays may involve. For example, the worst-performing machine (rank 1) with the *Random* scheme receives a bandwidth of a little lower than 600Kbps on average. We use the same way of presenting data in all

Schemes	NRU
DVMRP	1.00
Naive Unicast	2.62
Random	2.24
Narada	1.49

Table 2.1: Average normalized resource usage of different schemes

our comparison results.[1]

We wish to make several observations. First, the *Sequential Unicast* curve 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, *Narada* is comparable to *Sequential Unicast*. It is able to ensure that even the worst-performing machine in any run receives 1150 Kbps on average. Further, it results in consistently good performance, as indicated by the small standard deviations. Interestingly, *Narada* results 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 [67]. 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 600Kbps. Further, the performance of *Random* can be quite variable as indicated by the large standard deviation. We believe that this poor performance with *Random* is because of the inherent variability in Internet path characteristics, even in relatively well connected settings.

Figure 2.10 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, *Narada* is good at optimizing the overlay for delay. The worst machine in any run has an RTT of about 160 ms on average. Third, *Random* performs considerably worse with an RTT of about 350 ms for the worst machine on average. *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.

Table 2.1 compares the mean normalized resource usage (Section 2.3.2) of the overlay trees produced by the various schemes. The values are normalized with respect to the resource usage with DVMRP. 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 DVMRP. *Narada* can result in trees that make about 50% more use of resources than *DVMRP*. Further, *Naive 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.

2.3.4 Simulation Results

In this section, we study the performance issues using simulation experiments. We have investigated the effects of the following factors: (i) group size, (ii) fanout range, (iii) topology model; (iv) topology size.

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

Fanout Range

With Narada, each member in the data delivery tree has a degree that is dynamically configured based on the available bandwidth near a member. If a member has too many children, this could result in congestion near the member and a decrease in the available bandwidth. Narada can adapt dynamically to such a situation by detecting the fall in bandwidth and having children move away. However, given that our simulator does not consider Internet dynamics, we model the impact of this artificially by imposing restrictions on the degree. We do this using a parameter called the *fanout range*. 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 better delay performance. However, it could potentially result in higher stress on links near members. We vary the fanout range from $<2 – 4 >$ to $< 8 –16 >$ and study the performance impact.

Topology Model and Topology Size

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. [84] 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 [28]. 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 [27]. This data has been analyzed in [24] 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 paper. 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

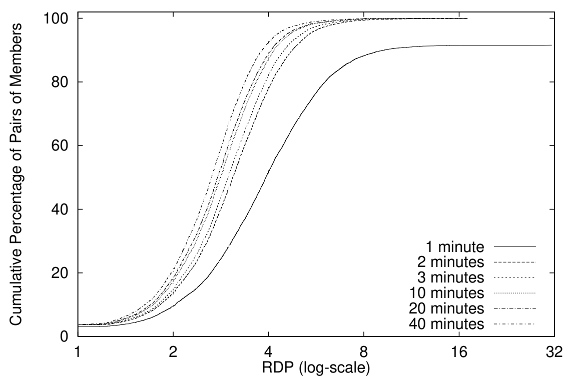

Figure 2.11: Cumulative distribution of RDP shown at various snapshots of the simulation. The minutes denote the time after the last join.

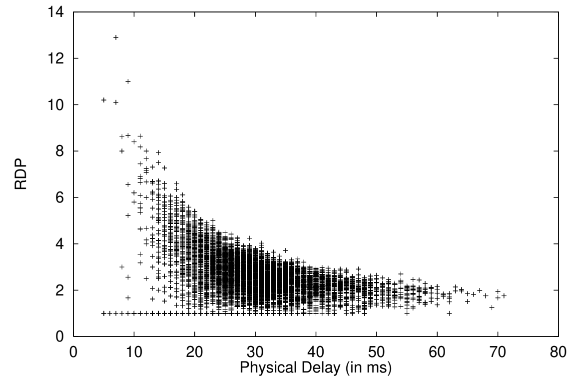

Figure 2.12: RDP vs. physical delay. Each point denotes the existence of a pair of members with a given physical delay and RDP

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.

Simulation Setup

Our simulation experiments are conducted using a locally written, packet-level, event-based simulator. The simulator assumes shortest delay routing between any two members. The simulator models the propagation delay of physical links but does not model bandwidth, queueing delay and packet losses. This was done to make our simulations more scalable.

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.

2.3.4.1 Results From a Typical Experiment

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>.

Figure 2.11 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

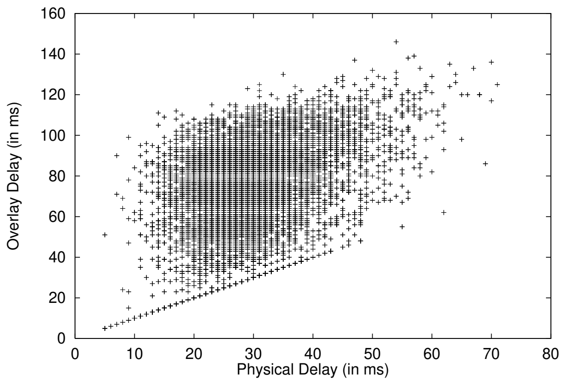

Figure 2.13: Overlay delay vs. physical delay. Each point denotes the existence of a pair of members with a given physical delay and overlay delay

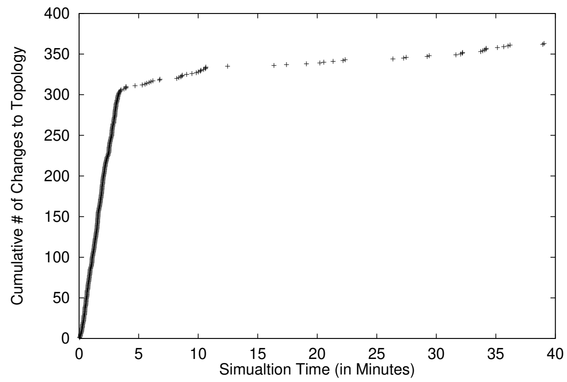

Figure 2.14: Cumulative number of virtual links added and removed vs. time

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 2.12. 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 delay between them along the overlay is not too high. This can be seen from Figure 2.13, 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 2.14 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 2.11 and indicates that the mesh quickly stabilizes into a good structure.

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 2.15. One of the members is picked as source at random, and we evaluate the stress of each physical link. Here, the

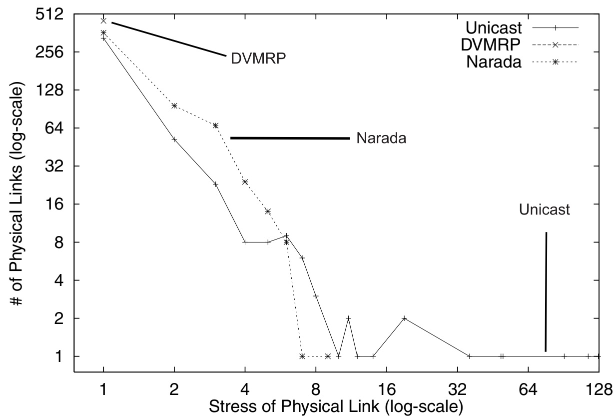

Figure 2.15: Number of physical links with a given stress vs. Stress for naive unicast, Narada and DVMRP

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 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.

2.3.4.2 Impact of factors on performance

We are interested in studying 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. By default, we used a Waxman topology with 1024 nodes and 3145 links, a group size of 128 and a fanout range of $<$3-6$>$ for all group members. For all results in this section, we compute each data point by conducting 25 independent simulation experiments and we plot the mean with 95% confidence intervals. Due to space constraints, we present plots of selected experiments and summarize results of other experiments.

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.

Figure 2.16 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

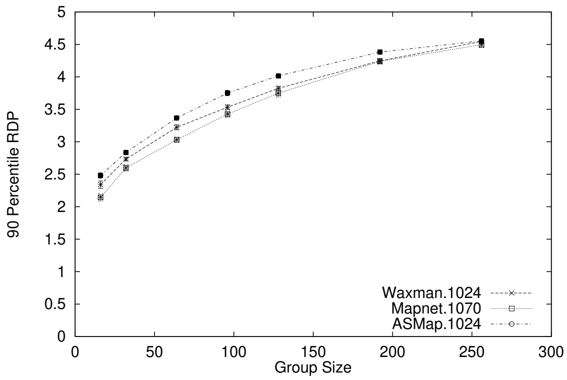

Figure 2.16: 90 percentile RDP vs. group size for topologies from three models

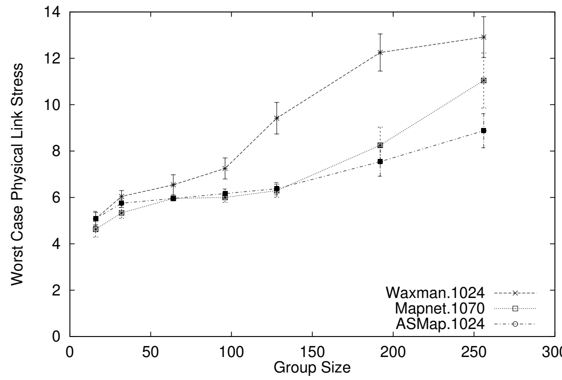

Figure 2.17: Worst case physical link stress vs. group size for topologies from three models

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 2.17 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 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 analyze this in detail in Section 2.3.5.

Figure 2.18 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. Figure 2.19 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 analyze the significance of this result in Section 2.3.5.

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 increase in topology

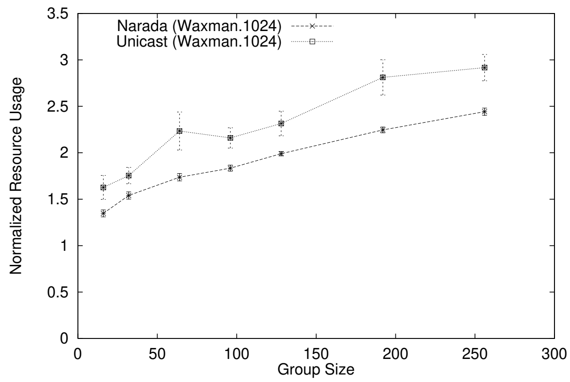

Figure 2.18: Effect of group size on NRU : Narada vs. naive unicast

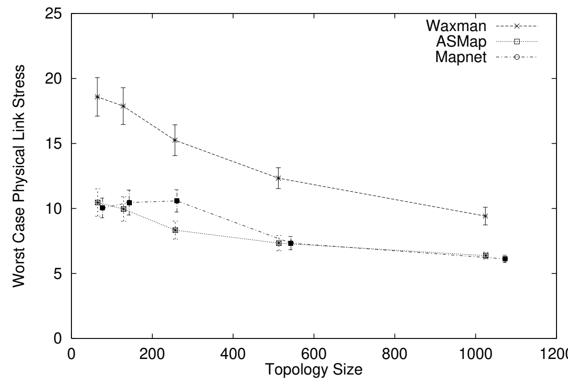

Figure 2.19: Worst case physical link stress vs. topology size for topologies from three models

size. We omit the plots due to space constraint.

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.

2.3.5 Results Summary

Overall our results suggest that End System Multicast is a promising architecture for enabling group communication applications on the Internet.

- *Bandwidth and latency*: Our Internet results demonstrate that End System Multicast can indeed meet the bandwidth requirements of applications while at the same time achieving low latencies. In Internet experiments, all hosts sustain over 95% of the source rate while seeing latencies lower than $80ms$. Our simulation results indicate that RDP is low across a range of topology models. For example, for a group size of 16, the 90 percentile RDP is less than 2.5. Even for group sizes of 128 members, the 90 percentile RDP is less than 4.

- *Stress*: Across a range of topology models, Narada results in a low worst case stress for small group sizes. For example, for a group size of 16, the worst case stress is about 5. While for larger group sizes, worst case stress may be higher, it is still much lower than unicast. For example, for a group of 128 members, Narada reduces worst case stress by a factor of 14 compared to unicast.

We hypothesize that worst case stress on the Internet is lower than seen in simulations. The largest topologies that we use in our simulations (around 1000 nodes) are still orders of magnitude smaller than the Internet. Consequently, the ratio of the group size to topology size, which we term *density*, is much higher in simulations than in actual practice. Our simulations indicate that higher group density results in higher worst case link stress. This can be deduced from Figures 2.17 and 2.19, where we observe that the worst case stress increases with group size and decreases with topology size. We hypothesize that an increase in group density increases the probability that an internal physical link could be shared by multiple uncorrelated virtual links. The links are uncorrelated in the sense that they connect distinct pairs of end systems. This could increase worst case stress with Narada because Narada is only able to regulate fanout of members and consequently can only control stress of links near member and not stress of internal physical links. For the range of group sizes we consider, we expect that the density ratio is much lower on the Internet and thus we expect lower link stress.

- *Resource usage*: Our Internet results demonstrate that Narada may incur a resource usage that is about $30 - 50\%$ higher than with DVMRP, while it can improve resource usage by $30 - 45\%$ compared to naive unicast. Again, our simulation results are consistent with our Internet results, and indicate that the performance with respect to this metric is good even for medium sized groups. The resource usage is about $35 - 55\%$ higher than with DVMRP for group sizes of 16 members, and about a factor of two higher for group sizes of 128 members. Further, we believe that the performance in resource usage may be even better if we consider clustered group members.

2.4 Related Works

Parallel to our End System Multicast work, Yoid [29], Scattercast [14], and Overcast [36] share the same motivation to move multicast functionality from routers to end systems. We are among the first to build a working protocol and conduct detailed evaluation in simulation and Internet testbeds to study the feasibility of this architecture.

Despite the common motivation, these projects approach the new architecture with different assumptions about the end systems. Yoid emphasizes end systems as user machines at the application endpoint, while Scattercast and Overcast argues that end systems should be part of the network infrastructure. In our view, end systems are generic entitities located at the "edge" of the network, and both instantiations of end systems have merit. In Section 2.6, we discuss the characteristics and performance implications of different instantiations of the end systems.

As the research community has begun to acknowledge the importance of overlay based architectures, many excellent research works have proposed new techniques to improve the *overlay multicast protocols*. In the following we categorize them by the structure in which the protocols construct. Note that we have leveraged some of these techniques to build the broadcast system and the incentive mechanism, described in Chapter 3 and Chapter 4.

- *Single Tree Structure*: Protocols like Yoid [29], BTP [32] and Overcast [36] construct trees directly - that is, members explicitly select their parents from among the members that they know, and data is sent along the tree to the children iteratively. Overcast targets for reliable *single-source* multicast applications, where real-time data delivery is not an objective. Thus the tree is optimized for good bandwidth from the source to each receiver, but not necessary low latency. Yoid supports *multiple source* applications, and constructs a single shared tree for all sources.

- *Routing over a Sparse Mesh*: Scattercast [14] proposes a self-organizing protocol called Gossamer. Gossamer is similar to Narada in that it construct trees in a two-step process: they first construct efficient meshes among participating members, and in the second step construct spanning trees of the mesh using well known routing algorithms. The key difference is that Gossamer relies on centralized rendezvous points for repairing mesh partition. Although this assumption significantly simplifies the partition recovery mechanisms in Gossamer, the members of the mesh could become partitioned from each other in the event of failure of all rendezvous points.

  Our choice of a mesh-based approach has been motivated by the need to support multi-source applications, such as video conferencing. Since video broadcast is also a single source application, we have taken ideas from Overcast, and modified the Narada protocol to construct a tree directly. The details are presented in 3.2.1.

- *DHT and Logical Structures*: Delaunay Triangulations [43], CAN [60], Bayuex [85], and Scribe [12] assign logical addresses to members from an abstract coordinate space, and neighbor mappings are based on these logical addresses. For example, CAN assigns logical addresses from cartesian coordinates on an n-dimensional torus. [43] assigns points to a plane and determines neighbor mappings corresponding to the Delaunay triangulation of the set of points. Thus, multicast routing within this logical space is implicitly defined, saving the overhead of maintaining the routing states. The key benefit is that the protocols scale better to larger group sizes. However, in contrast these protocols impose rules on neighbor relationships that are dictated by logical addresses assigned to hosts rather than performance. This may involve a performance penalty in constructed overlays and could complicate dealing with dynamic metrics such as available bandwidth.

- *Hierarchy and Clustering*: Nice [4] and Kudos [35] achieve better scaling properties than Narada by organizing members into hierarchies of clusters. Kudos constructs a two level hierarchy with a Narada like protocol at each level of the hierarchy. NICE constructs a multilevel hierarchy, and does not involve use of a traditional routing protocol. A concern with hierarchy-based approaches is that they complicate group management, and need to rely on external nodes to simplify failure recovery.

- *Multiple Disjoint Trees*: CoopNet [53] and SplitStream [11] improve data resiliency by sending data along multiple disjoint trees. So even if a subset of the hosts fail, most hosts will still receive some portion of the data. Such a technique can be combined with appropriate data encodings to improve application performance. For example, applications can use erasure coding for bulk data transfer [7], or multiple description codec for streaming media [50]. In Chapter 4, we leverage this technique to incorporate the incentive semantic into a multiple disjoint trees protocol.

- *Centralized Structure Maintenance*: While Narada advocates for self-organization on an overlay network, ALMI[56] and CoopNet [53] propose to construct overlays in a centralized fashion. In ALMI, end systems periodically report network performance to a session controller. The session controller computes a minimum spanning tree and informs the parent-child relationship to end systems.

The MBone [10] and 6Bone [37] are popular existing examples of overlay networks. However, these are statically configured in a manual and ad-hoc fashion. Narada, on the other hand, strives for a self-configuring and efficient overlay network. Internet routing protocols are self-configuring. The most striking difference between Narada and standard routing protocols is that while the latter work on a fixed physical topology, Narada alters the very topology over which it routes data. Routing protocols merely route around a link that has failed and have no notion of dynamic adding or dropping of links. Narada might dynamically add links to ensure connectivity of the virtual topology, and drop links it perceives as not useful.

Self-configuration has been proposed in other contexts. AMRoute [8] allows for robust IP Multicast in mobile ad-hoc networks by exploiting user-multicast trees. Several reliable IP Multicast protocols [42, 44, 83] involve group members self-organizing into structures that help in data recovery. Adaptive Web Caching [48] is a self-organizing cache hierarchy. The key feature that distinguishes Narada from these protocols is that Narada does not assume a native multicast medium - AMRoute assumes a native wireless broadcast channel, while all other protocols assume the existence of IP Multicast. Self-configuration in the absence of such a native multicast medium is a much harder problem.

2.5 Summary

We have made two contributions in this chapter. First, we have shown that for small and medium sized multicast groups, it is feasible to use an end system overlay approach to *efficiently* support all multicast related functionality including membership management and packet replication. The shifting of multicast support from routers to end systems, while introducing some performance penalties, has the potential to address most problems associated with IP Multicast. We have shown, with both simulation and Internet experiments, that the performance penalties are low in the case of small and medium sized groups. We believe that the potential benefits of transferring multicast functionality from end systems to routers significantly outweigh the performance penalty incurred.

Second, we have proposed one of the first self-organizing and self-improving protocols that constructs an overlay network on top of a dynamic, unpredictable and heterogeneous Internet environment without relying on a native multicast medium. We also believe this is among the first works that attempts to systematically evaluate the performance of a self-organizing overlay network protocol and the tradeoffs in using overlay networks. Further, we believe that the techniques and insights developed in this chapter are applicable to overlay networks in contexts other than multicast.

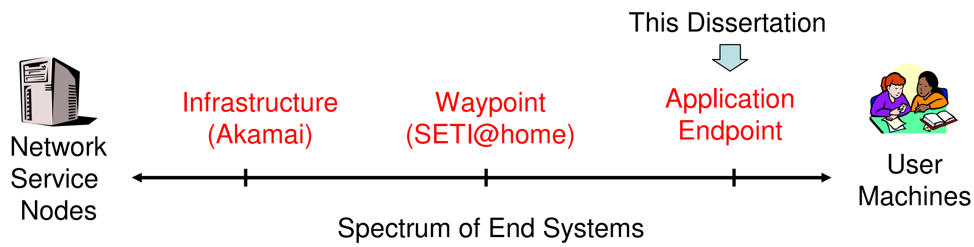

Figure 2.20: A spectrum of end systems on the Internet. This dissertation explores one of the spectrum where end systems are user machine.

2.6 Discussion: What is an End System?

End System Multicast advocates for an architecture where data replication should be performed not by the routers, but by the end systems. But *what is an end system?* It turns out there is a wide spectrum of end systems on the Internet as shown in Figure 2.20, and the choice of end systems can have significant implication on the characteristics and performance of this architecture.

This dissertation explores one end of the spectrum, where end systems are the user machines who participate in the broadcast. We call this the *application endpoint* architecture instantiation. The literature also refers to this architecture as a *peer-to-peer* (P2P) architecture. We use these two terms interchangeably in this dissertation.

In the following we review two other common types of end systems and their respective architecture instantiations. The end systems differ by the *incentive* in participating in the system, the *persistency* in staying in the system, how well they are *provisioned* in network bandwidth, and the degree of *sharing* among different multicast groups. We discuss the tradeoff and justify our choice of the architecture instantiation.

2.6.1 End Systems as Infrastructure Nodes

The other end of the spectrum is the *infrastructure-based* architecture instantiation. In this architecture instantiation, end systems consist of highly provisioned and highly available infrastructure nodes. These infrastructure nodes run the End System Multicast protocol and offer multicast as an application-level service to the *clients*. In our application, clients are viewers (or application endpoints) who want to receive the video broadcast. The clients attach themselves to nearby infrastructure nodes and receive data using plain unicast (or perhaps intra-domain LAN Multicast, or another level of application multicast).

This architecture instantiation has been adopted by third-party value-added service providers such as Akamai [2] and Real Broadcast Networks [61]. There are many advantages in this architecture instantiations:

- *Better Performance*: The infrastructure-based architecture can likely deliver better multicast performance than an application endpoint architecture for two reasons. (i) Infrastructure nodes are *persistent*. This helps to avoid the transient data loss associated with ancestor nodes failure or leaving the group. (ii) Infrastructure nodes are *well-provisioned* with high and stable bandwidth. This helps to avoid performance loss due to network congestion near the parent nodes. On the other hand, an application endpoint architecture must cope with performance loss due to group dynamics and network congestion.

- *Simpler Protocol*: Because the infrastructure nodes are persistent and well-provisioned, the overlay protocol is *simpler to construct* and can *scale to larger group size*. In fact, earlier version of the Akamai system constructs a static overlay because the nodes are very stable. Moreover, the structure can sustain high fanout because each infrastructure node can support a large number of clients simultaneously. This greatly reduce the number of nodes to manage when group size is large. An smaller overlay structure is easier to maintain. In contrast, peers in application endpoint architecture have limited bandwidth and short join duration. This results in a dynamic overlay structure that grows with group size, which is much difficult to maintain in a distributed fashion.

- *Inherently Cooperative*: Infrastructure nodes are inherently cooperative in participating in the broadcast system because they owned by a central entity. However, peers in application endpoint architecture may not. With the prevalence of free-riding [1] reported in peer-to-peer file sharing applications, it is not clear whether peers in the video broadcast applications would be cooperative in contributing bandwidth. Chapter 4 is devoted to address this issue.

- *More Secure*: Infrastructure nodes are typically centrally managed and the service is explicitly invoked, thus they are easier to secure. On the other hand, peers in application endpoint architecture may be malicious or non-cooperative. For example, malicious nodes could disrupt the consistency of the distributed structure by sending bogus control data, and they could disrupt the data path of a large subset of nodes by selectively attacking the replicating nodes close to the source.

Despite the potential benefits with infrastructure-based architecture, the main concern is cost.

- *Cost*: The service provider incurs cost when the infrastructure nodes “touch” the broadcast data. This is unavoidable as they must replicate data among themselves, and serve data to clients. Moreover, the cost will increase with more viewers tuning in the broadcast. This cost must be *explicitly paid for* by some entities, such as the publisher, the viewers, or another third-party such as advertisers. For many broadcast events, there may not be an entity who wants to explicitly pay for this cost of replication.

  In the application endpoint architecture, the replication cost is *implicitly shared* among the peers. Most peers pay a flat-rate fee for a fixed bandwidth pipe to the Internet, which includes both upstream and downstream bandwidth. By replicating data in peers, this architecture implicitly delegates the replication cost to individual peers. This cost sharing is also *self-scaling*: peers expect to replicate a fixed amount data regardless of group size, as the bandwidth resource increases proportionally with more peers in the system. The inherit cost-sharing model nicely matches the incentive for both the publishers and the viewers. By delegating the task of replication to the peers, publishers can avoid costly bandwidth provisioning to support a large number of viewers. The peers have an incentive to help the publisher in exchange for enjoying the video that might otherwise not be available.

We believe that application endpoint is the most suitable architecture for the objective of this dissertation: to provide broadcasting as a *commodity* service. The lowered cost justifies the potential tradeoff with worse performance, higher complexity, and degraded security. However, we believe this architecture instantiation is *not* suitable for all circumstances. In particular, infrastructure-based solution fits well with publishers who demand high performance and security.

Our deployment experience in Chapter 3 is a testimony of this architecture tradeoff. The reason why we can successfully deploy the service within several months is because there is no "cost" (i.e. money) for either the publisher or the viewers. However, we have to address many of the issues inherited to this architecture. Our experience indicates that with careful engineering of the protocol, performance is acceptable in the presence of group and network dynamics. Moreover in Chapter 4, we address the issue of collaboration by providing an incentive mechanisms that are compatible with both the publisher and the viewers. Security is one outstanding challenge we have not addressed, and we discuss this in the future work section (Section 5.3).

2.6.2 End Systems as Voluntary Waypoints

There is a trend of increasing altruism on the Internet. People with idle resources, such as bandwidth, CPU, and storage, sometimes donate them to a *cause* voluntarily. Large-scale, voluntary contribution of resource has been observed in many (peer-to-peer) systems, such as SETI@home [70] and PlanetLab [58]. We argue that broadcast applications "naturally" brings out altruism in people. Live broadcast is typically associated with a specific event people care about. For example, for the application example we sketched above, a broadcaster of a seminar series may request members of its community in various universities to join the broadcast and leave their machines on. We call such hosts *waypoints*. These waypoints hosts have characteristics similar to infrastructure nodes: well-provisioned and stable. Application end-point architectures can gracefully leverage the resources of waypoints. In the absence of waypoints, the system provides performance comparable to a purely application end-point based solution: in the presence of waypoints, the performance can be enhanced.

Our system described in Chapter 3 can gracefully leverage these waypoint hosts. The system is *transparent* in a sense that it makes no distinction between waypoints and normal application end-points. Waypoints may join and leave the system, like any participating host. However, performance may be improved if waypoint is explicitly considered when constructing the overlay. For example, they should be placed closer to the source host as they are more stable and can sustain higher fanout. This is beyond the scope of this dissertation. For some real broadcast events, we use the PlanetLab hosts as transparent waypoints to ensure the success of the deployment.

Chapter 3

Broadcast System

The process of this dissertation research is like a building a house. If the previous chapter is about drawing an architecture blueprint and creating a model house, then this chapter is about building the actual house, inviting people to live in there, and learning the design implication by observing how the occupants interact with the house.

To validate the End System Multicast architecture proposed in Chapter 2, we have implemented and deployed a video broadcast system, where data replication is performed completely by the viewers. To date, the system has been in operation since August 2002. It has been used to successfully broadcast many events including academic conferences such as ACM SIGCOMM and SOSP, DARPA Unmanned Vehicle Grand Challenge, CMU Commencement, and CMU SCS Distinguished Lectures. These broadcasts have reached thousands of users in home, academic and commercial environments, spread over four continents [16]. The deployment experiment has been insightful. It demonstrates the great potential of using the application endpoint architecture to provide cost-effective video streaming over the Internet. Moreover, the experience has led us to identify first-order issues that are guiding our research and are of importance to any overlay multicast protocol or system.

This chapter presents the details of the system implementation and the performance results. Section 3.1 presents an overview of the system design. Section 3.2 provides the detail descriptions of each component in the design. Section 3.3, 3.4, and 3.5 present the deployment experience, analysis methodology, and performance analysis of our system. Section 3.6 presents key design lessons learned from the experience.

3.1 System Design Overview

Building an operational software system in a research setting is a delicate balanced act. On the one hand, the system must be robust and easy to use to satisfy the needs of the publishers and the viewers. We can gain research insight only if people use our system. On the other hand, building such an operational system seems to require a comprehensive engineering effort with significant manpower. This is a luxury we as university researchers do not have.

Thus our system is designed to satisfy the needs of the users with minimal engineering effort. Section 3.1.1 identifies design objectives that are important to the publishers. Section 3.1.2 presents our design choices to simplify the engineering efforts. Section 3.1.3 gives an overview of the system from the operations perspective.

3.1.1 Design Objectives

Our broadcast system is designed to achieve the following three objectives:

- *Easy to Use*: The system should be easy to use by the publishers and the viewers. This encourages more people to use the system, which in turn generates valuable usage traces for research analysis. We measure the ease of use by the *time* publishers and viewers must invest to broadcast and join an event. With our current system, typical publishers take an hour or so to setup the system and a few minutes to start each broadcast event. Most viewers takes a few minute to install our software and a few seconds to join the broadcast. This is achieved by a Web portal to streamline the process of event broadcasting.

- *Ubiquitously Accessible*: We learn that one of the essential requirements for publishers is that the video broadcast must be accessible to *any* perspective viewers. Internet users are very heterogeneous in their connectivity constraints, network speed, and software/hardware platforms. If we do not support, let's say, Linux or Mac users, they complain loudly to the publishers.

  In contrary to our early expectation, it takes tremendous effort (both in research and in engineering) to make the broadcast system ubiquitously accessible. On the research front, we have designed the system to support hosts behind NAT and a subset of hosts behind firewall. Moreover, the system provides different video bitrate to users with different network speed. Users can get reasonable quality video if they have at least DSL/cable network speed. On the engineering front, we have ported our code to run on Windows, Linux, and Mac.

- *Robust*: Unlike a research prototype, an operational system must be robust to be used in a large scale. Failure would significantly deter event organizers and limit future adoption of our system. One consequence is that it is critical to adopt robust, stable and well-tested code – a performance refinement that seemed trivial to incorporate may take months to actually be deployed.

  To build a robust code, we have created extensive testing environments and methodological testing procedures. However, it is difficult to emulate the environment of a real broadcast event in testbed, and we make the real broadcast events as the ultimate testing environment. To this end, we build a robust logging infrastructure to gather logs of user performance and analyze the logs post-mortem. During the run-time, we also have a monitoring system to collect simple real-time statistics on components of the system, and attempt to automatically recover if they fail.

3.1.2 Design Choices

In making a design decision, we are often faced with a difficult choice between simplicity and feature/performance. In the first iteration of the design, the choice is always toward simplicity. There are two reasons behind this choice. First, we want to minimize the engineering effort and quickly reach an operational state. Second, without operational experience, it is difficult to gauge the importance of a new feature or a better design. As we gained deployment experience, we have iteratively revised and improved our system design.

- *Leverage Off-the-Shelf Components*: As a first iteration, we leverage as much off-the-shelf components as possible to rapidly prototype an operational broadcasting system. For example, we have leveraged commercially available media players (i.e. QuickTime) and audio/video codecs (i.e. Sorenson). Moreover, we resort to TCP for congestion control, instead of other proposed real-time streaming protocols such as TFRC, for reasons of code robustness and stability.

  Some of these off-the-shelf components are not well suited for this broadcast system based on an End System Multicast architecture. Since customization of these off-the-shelf components are nearly impossible, the system must bear certain performance penalty. For example, we adapt a simple media codec with single bitrate video encoding, which incurs overhead when adapting for receiver heterogeneity. Moreover, prioritization of audio data over video data is less effective with TCP sockets. Ongoing works aim to replace these components for better performance and greater flexibility in user-interface customization. For example, Chapter 4 assumes a more sophisticated video codec (i.e. MDC) to express incentive policies of peers in a broadcast system.

- *Design for Today's Environment*: Our deployment horizon is in months and not in years. Thus the system must be engineered to work in today's environment. One example is the support for hosts behind NAT and firewalls. Communication is infeasible between certain pairs of hosts (for example, two typical NATs) and restricted between other pairs of hosts (for example, a host outside a firewall cannot initiate a TCP connection to a host inside a firewall). The problem with NATs might seem to be temporary or one that would vanish with widespread use of IPv6 [23] or Universal Plug and Play (UPnP) [76]. However, these solutions are at least a few years off, and NATs and firewalls are going to be prevalent for the foreseeable future. Good systems design allows for incremental deployment and we are motivated to directly address support for such hosts.

- *Simple Protocol Design*: The protocol used in the system starts from the Narada protocol code-base in Section 2.2. The major modification is from a mesh structure (suitable for multi-source conferencing applications) to a tree structure (suitable for single-source broadcast application). We refrain from making potential performance enhancements in consistent with code simplicity. As an example, there is no explicit coordination among peers to repair the tree structure when one overlay link sees congestion. Also peers do not explicitly repair packets when packet is lost along the tree structure. In light of the deployment experience, they are not the first order performance concerns. However, future work should implement them to continue improving the system performance.

- *Assume Altruistic Viewers and Harness Resource Efficiently*: There is a concern early on whether peers in an application endpoint architecture have enough bandwidth resources to replicate high bandwidth video streams. To this end, we have made two design and deployment decisions. (i) We do not specify to the viewers how much bandwidth they will contribute

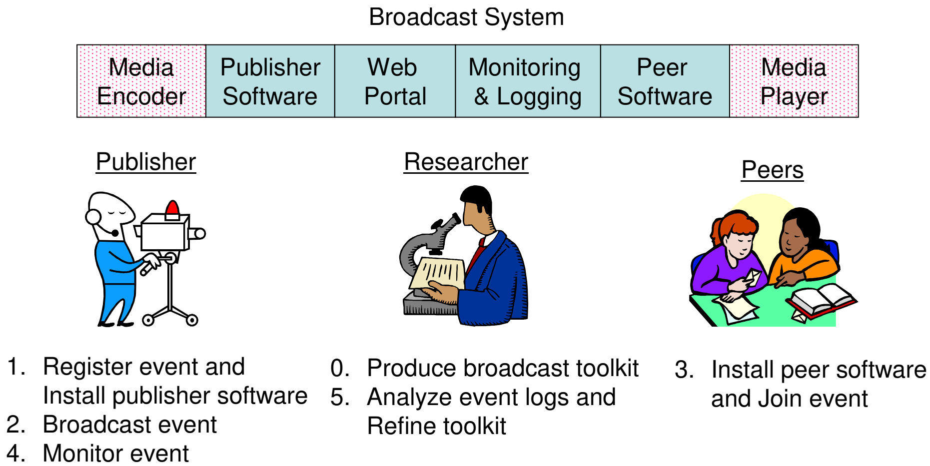

Figure 3.1: An overview of the broadcast system components. In a typical broadcast, these components are invoked in a sequence listed in this figure. A flow representation of the component invocation is shown in Figure 3.2

to the system. By default, the our software harnesses up to six times the source rate from each peer, and assume peers are obedient running the software. In Chapter 4, we relax this assumption where peers are strategic when contributing their bandwidth. (ii) In designing to support hosts behind NAT and firewall, we choose to enable them to be bandwidth contributors (parents). Support for such hosts could have been simpler had we limited their participation only as bandwidth consumers (children). We make this choice at the cost of additional design complexity.

3.1.3 System Overview

Figure 3.1 presents an overview of the system from the operations perspective. The broadcast system is composed of six software components. These components are run by the publishers, viewers, and us (i.e. researchers at CMU). We currently run two components (Web portal and monitoring/logging) to simplify the tasks of publishers, and to get logs for research. Ongoing works allow the publishers or other third parties to invoke these components independently as needed.

- *Media Encoder and Player*: The publisher runs media encoders, which encodes media signal (audio/video) into compressed digital streams. The viewers run media players, which decode the media streams and display the audio/video content at their computers. We leverage commercially available software for these two components.

- *Publisher and Peer Software*: The software components are invoked during the broadcast. These components are responsible for forwarding and receiving audio/video streams over the Internet. They collaboratively maintain an efficient overlay structure in a distributed fashion. The bulk of our research is in these software components.

- *Web Portal*: This is the rendezvous point for the publishers and the viewers. The Web portal provides a convenient interface for publishers to register broadcast events. The viewers can

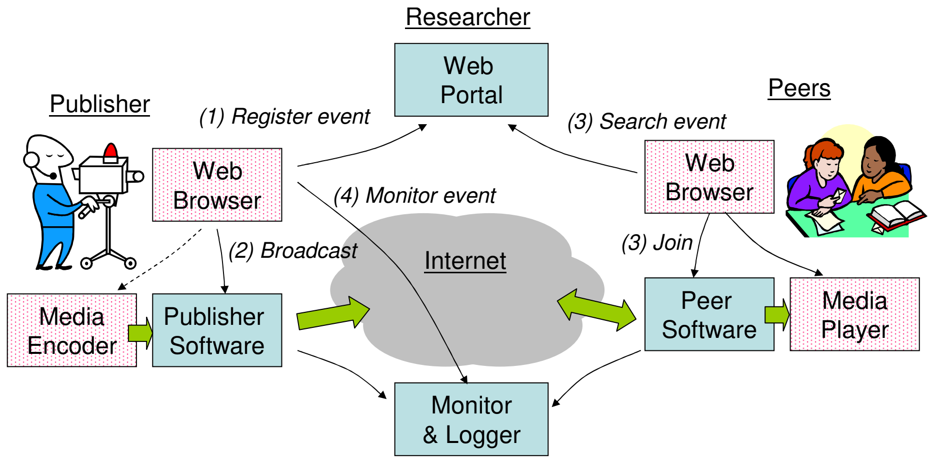

Figure 3.2: In a typically broadcast event, the system components are invoked in the four steps as shown in the figure.

search the event listing and join the events of their interests.

- *Monitoring and Logging*: There is a separate component to monitor the system in real-time. It tracks vital statistics about the events such as number of people, their loss, and positions in the tree. The statistics are accessible through the Web portal. They assure the event publishers of a successful ongoing broadcast, and help us debug the system when trouble arises. We also collect detail user performance logs in the background for post-mortem analysis.

The system components are typically invoked the following sequence depicted in Figure 3.2.

1. The publisher first registers the event to our Web portal via a Web browser. The portal assigns an event URL, which is essentially an application-level name of the event. The publisher then announces the URL to the perspective viewers separately (e.g. via email or newsgroup).

2. When the event starts, the publisher goes to the Web portal, and invoke the encoder and the publisher software with appropriate configuration. The configuration includes the encoding type (encoding format, bitrate, frame size) and the network information (encoder and source IP addresses and port numbers).

3. To join the broadcast event, a peer simply goes to the event URL. The URL provides further directions to install and invoke the software with appropriate configurations.

4. The publisher can monitor the event through the Web portal and shut down the event at the end.

The thick arrows in Figure 3.2 highlights the data path among the components. The encoder takes the media signal from a source (e.g. camera and microphone), converts into audio and video streams, and sends to the source host running the publisher software. Together with the peer software, the replicate the streams using an overlay multicast protocol. The peer software forwards a copy of the stream to the media player (running on the same machine) and display to the viewer. In

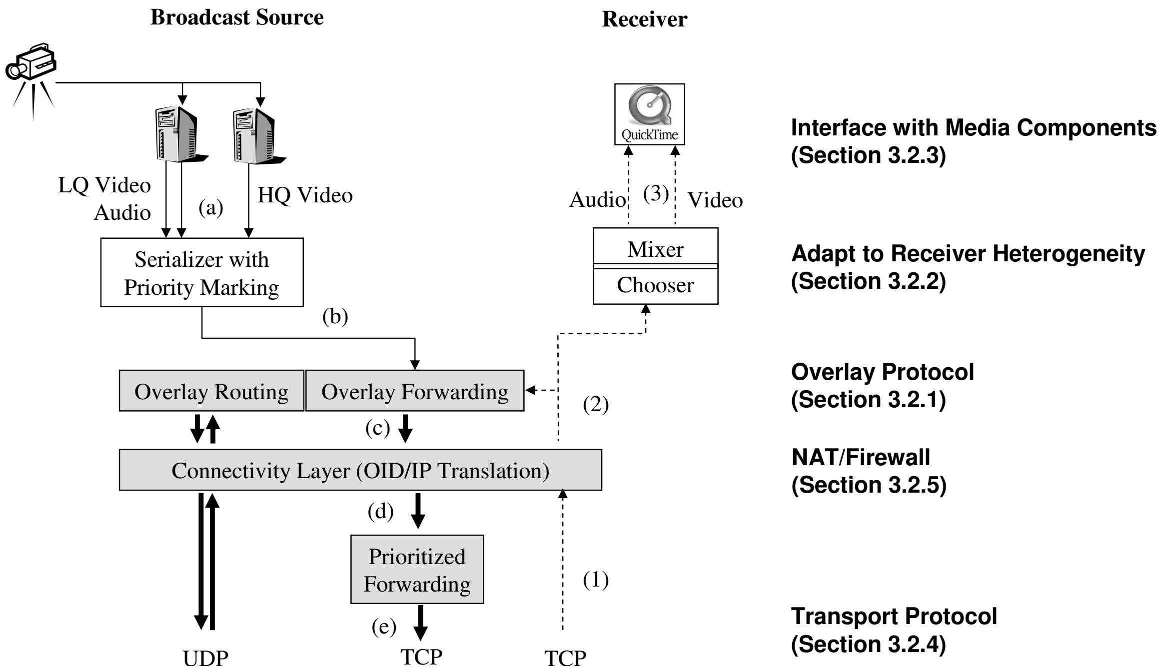

Figure 3.3: Block diagram of the architecture for the publisher software (left) and the peer software (right). Shaded blocks are shared by both components. Arrows indicate data flow.

addition, the peer software periodically sends performance data to the monitor and log servers for on-line and off-line analyses.

3.2 Detail System Descriptions

The bulk of the research and engineering is on the publisher and viewer software components. This section describes the two components in detail. We break them down into five functional blocks. Figure 3.3 shows the block diagrams. The shaded blocks are shared by both the publisher and the viewer software. We note that many of the functionality are shared, including the overlay protocol and NAT/firewall traversal.

These functional blocks are organized in layers. The top layer interacts with the media encoder for the publisher and the player for the viewer (Section 3.2.3). The next layer serializes and prioritizes the media streams to cope with receiver heterogeneity and network congestion (Section 3.2.2). The overlay protocol layer maintains the distributed state of parent/child peers, where they are identified as a logical name (Section 3.2.1). The next layer maintains the binding between the logical name and the physical address, which is used to traverse NAT/firewall (Section 3.2.5). The lowest layer interacts with the TCP/UDP sockets provided by the host operating systems (Section 3.2.4).

This section also describes the detailed design of other software components, including logging and monitoring (Section 3.2.6), and Web portal and user interface (Section 3.2.7).

3.2.1 Overlay Protocol

Because our application is single-source, the protocol builds and maintains an overlay tree in a distributed fashion. The tree is optimized primarily for bandwidth, and secondarily for delay. To maintain good performance, hosts monitor their receiving bandwidth and switch parents if they observe a performance drop. When selecting parents, the child looks for one that can provide better performance. Each node also maintains a degree bound of the maximum number of children to accept. A group management protocol is used to learn about other members that could be used as parents. We highlight some of the key components below.

- *Group Management:* New hosts join the broadcast by contacting the source and retrieving a random list of hosts that are currently in the group. It then selects one of these members as its parent using the parent selection algorithm. A key necessity for a self-organizing protocol is knowing other members that could be contacted as new parent candidates if the current parent leaves or provides poor performance. Each member maintains a partial list of members, including the hosts on the path from the source and a random set of members, which can help if all members on the path are saturated.

  To learn about members, we use a gossip protocol adapted from [63]. Each host A periodically (every 2 seconds) picks one member (say B) at random, and sends B a subset of group members (8 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. Finally, members are deleted if its state has not been refreshed in a period (5 minutes). This helps to expire information about members that have left the group.

- *Handling Group Membership Dynamics:* Dealing with graceful member leave is fairly straightforward: hosts continue forwarding data for a short period (5 seconds), 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 liveness.

- *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.

  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. While our initial implementation did not consider loss rate as a metric, we found it necessary to deal with variable-bit-rate streams, as dips in the source rate would cause receivers to falsely assume a dip in performance and react unnecessarily. Thus, our solution avoids parent changes if no packet losses are observed despite the bandwidth performance being poor.

- *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 (30 in our implementation). The probing is biased toward members that have not been probed or have low delay. Each host B that responds

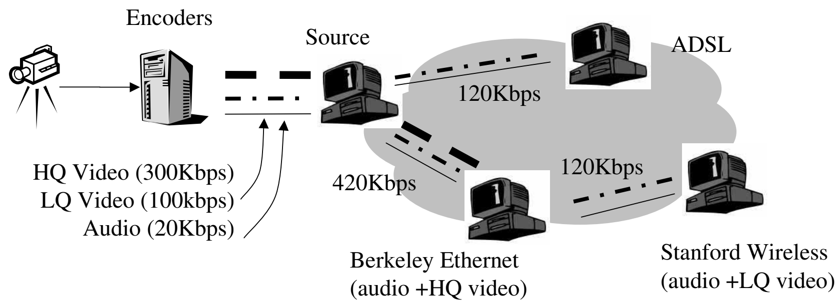

Figure 3.4: Single overlay approach to host heterogeneity.

to the probe provides 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 to prevent routing loops. The probe also enables A to determine the round-trip time to B.

A waits for responses for 1 second, then eliminates those members that are saturated, or who are its descendant. It then evaluates the performance (throughput and delay) of the remaining hosts if it were to choose them as parents. If A does not have bandwidth estimates to potential parents, it picks one based on delay. Otherwise, it 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. History of past performance is maintained so if A has previously chosen B as parent, then it has an estimate of the bandwidth of the overlay link $B - A$. A then evaluates how much improvement it could make if it were to choose B.

There are two conditions under which B can considerably improve performance. First, performance is improved if the estimated application throughput is high enough for A to receive a higher quality stream (see the multi-quality streaming discussion in Section 3.2.3). Second, if B maintains the same bandwidth level as A's current parent, but improves delay. This heuristic attempts to increase the tree efficiency by by making hosts move closer to one another.

- *Degree Bound Estimation:* In order to assess the number of children a parent can support, we ask the user to choose whether or not it has at least a 10 Mbps up-link to the Internet. If so, we assign such hosts a degree bound of 6, to support up that many number of children. Otherwise, we assign a degree bound of 0 so that the host does not support any children. We have been experimenting with heuristics that can automatically detect the access bandwidth of the host, but this turns out not to be straightforward. We discuss this further in Section 3.6.

3.2.2 Support for Receiver Heterogeneity

Internet hosts are highly heterogeneous in their receiving bandwidth, thus a *single-rate video coding* scheme is not the most appropriate. Various streaming systems have proposed using scalable coding techniques such layered coding or multiple description coding (MDC) in their design [47, 53, 11], however these technologies are not yet available in commercial media players. To strike a balance between the goals of rapid prototyping and heterogeneous receiver support, in our system, the source encodes the video at multiple bit-rates in parallel and broadcasts them simultaneously, along with the audio stream, through the overlay as shown in Figure 3.4. We run unicast congestion control on the data path between every parent and child, and a *prioritized packet forwarding* scheme is used to exploit the available bandwidth. That is, 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. Thus, audio is highly protected. 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 congestions or poor parent choices, as long as the lower quality video stream is received, a legible image can still be displayed. We note that while this design involves some overhead, it can be seamlessly integrated with layered codecs if available.

Much of the deployment experience reported in our work uses TCP as the congestion control protocol. We implement priority forwarding by having parents in the overlay tree maintain a fixed size per-child priority buffer. Packets are sent in strict priority and in FIFO order within each priority class. If the priority buffer is full, packets are dropped in strict priority and in FIFO order (drop head). The priority buffer feeds the TCP socket, and we use *non-blocking write* for flow control. Note that once packets are queued in kernel TCP buffers, we can no longer control the prioritization. While we were aware of this limitation with using TCP, we were reluctant to employ untested UDP congestion control protocols in actual large scale deployment. Our subsequent experience has revealed that while the choice of TCP has only a minor hit on the performance of the prioritization heuristics, a more first-order issue is that it limits connectivity in the presence of NATs and firewalls. Faced with this, we have begun incorporating TFRC [26], a UDP-based congestion control protocol, into the system.

To prevent frequent quality switches that could annoy a user, we adopted a damping heuristic. Here, we aggressively switch to lower quality when high quality video has consistent loss for 10 seconds, and conservatively switch to higher quality when no loss is observed in the higher quality video stream for at least 50 seconds. Dynamically switching video qualities required us to implement an RTCP mixer[69]. When video qualities are switched, the mixer ensures the outgoing video stream to QuickTime is (i) masked as one contiguous stream; and (ii) time synchronized with the audio stream. One limitation in our current implementation is that if a host is displaying a low quality stream, the parent still forwards some data from the high quality stream. We are currently refining the implementation by adding heuristics to have the child unsubscribe from the higher quality stream, and periodically conduct experiments to see when network condition has improved so that it can start receiving the high quality stream.

3.2.3 Interface to Media Components

We use *QuickTime* [59] as the media player in our system because it is widely available and runs on multiple popular platforms. We use *Sorenson 3* [72] and MPEG4, both of which are supported by QuickTime, as video codecs. To support receiver heterogeneity, the source encodes the video at two target bit-rates (100 Kbps and 300 Kbps), and the audio at 20 Kbps. We empirically determine the suitable encoding rates by experimenting with various encodings of conference talks. We find that a frame size of 640x480 is necessary to read the words on the slides. A minimal rate of 100 Kbps yields watchable, 5 frames per second video motion. A rate of 300 Kbps produces good video quality with 15 frames per second. To hide from the media player the fact that the overlay parent changes over time, we direct the player to a fixed *localhost:port* URL which points to the overlay proxy running at the same host. The overlay proxy handles all topology changes and sends data packets to the player as though it were a unicast streaming media server.

3.2.4 Transport Protocol

We distinguish between two types of message exchanged among peers: *control messages* and *data messages*. Peers exchange control messages to maintain the overlay structure, and the traffic volume is low, ranging from 6-20Kbps. On the other hand, the traffic volume for data messages is high, as they carry the audio/video streams. For example, if a parent forwards data to 6 children, each at a source rate of 420Kbps, the data rate is up to 2.5Mbps.

We use UDP to exchange control messages between two peers. Datagram delivery is prompt but not reliable. Prompt delivery ensures accurate delay estimates and good consistency for maintaining a distributed structure. Control messages can tolerate loss, as they carry soft state which are refreshed periodically.

Since data messages have high traffic volume, we choose to apply congestion control. This helps to avoid inflaming network administrators and/or interrupting other applications the viewers may be running. We experiment with two congestion control protocols, and select TCP over TFRC [26]. TFRC is a UDP-based congestion control protocol optimized for streaming media. Compared to TFRC, TCP is not ideal for real-time streaming because (i) the system loses control of the data once it is queued in the kernel buffer, and (ii) packet retransmission may be ineffective because it is too late. Despite the performance drawbacks, TCP is chosen because it is widely available and well-tested, a conservative choice in a large-scale Internet deployment.

To implement priority forwarding (for receiver heterogeneity) with TCP, each parent maintains a fixed size per-child priority buffer. The priority buffer sits above the TCP socket, and uses non-blocking write for flow control. Packets are sent in strict priority and in FIFO order within each priority class. If the priority buffer is full, packets are dropped in strict priority and in FIFO order (drop head). Note that once packets are queued in kernel TCP buffers, we can no longer control the prioritization. Our subsequent experience has revealed that while the choice of TCP has only a minor hit on the performance of the prioritization heuristics, a more first-order issue is that it limits connectivity in the presence of NATs and firewalls. Faced with this, we have begun incorporating TFRC [26], a UDP-based congestion control protocol, into the system.

In summary, our system uses UDP to transport control messages between two peers, and TCP for data messages.

3.2.5 NATs and Firewalls

Our initial prototype did not include support for NATs and firewalls. We were motivated to address this as we consistently needed to turn down a good fraction of the viewers in our early broadcasts for the lack of such support. NATs and firewalls impose fundamental restrictions on pair-wise connectivity of hosts on the overlay. In most cases, it is not possible for NATs and firewalls to communicate directly with one another. However, there are specific exceptions, depending on the transport protocol (UDP or TCP), and the exact behavior of the NAT/firewall. Adopting the classification from

TCP Transport (rows: Child; columns: Parent)

	Public	NAT	Firewall
Public	✓	✓	✓
NAT	✓	★	✗
Firewall	✓	✗	★

UDP Transport (rows: Child; columns: Parent)

	Public	NAT	Firewall
Public	✓	✓	✓
NAT	✓	★ ?	?
Firewall	✓	?	★ ?

Figure 3.5: Connectivity Matrix. $\surd$ means connectivity is always possible. X means the connectivity is never possible. ? means connectivity is possible for some cases of NAT/firewall and $\star$ means connectivity is only possible if the hosts are in the same private network.

STUN [65], *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. Many hosts can address a host behind a full cone NAT using the same port number. In contrast, *Symmetric NATs* allow incoming packets only from the host that it has previously sent a packet to. Different hosts address a host behind a symmetric NAT using different port numbers. Table 3.5 characterizes these restrictions for the different transport protocols, where columns represent parents and rows represent children. For example, communication is not possible between two NATed hosts using TCP unless they happen to be in the same private network. In addition, "?" denotes that communication is possible using UDP between two NATed hosts if one of them is behind a *Full Cone NAT*. The *firewalls* which we refer to in Table 3.5 allow UDP packets to traverse in either direction. The system does not support firewalls that block UDP.

The primary goals in supporting NATs and firewalls are: (i) enable connectivity, a generic problem shared by many applications wishing to support these hosts and (ii) address protocol-specific enhancements to become "NAT/firewall-aware" to improve efficiency and performance. The detail of enabling connectivity for NAT and firewall hosts can be found in [16].

3.2.6 Logging and Monitoring

We implement a reliable logging infrastructure that automatically collect performance logs from all hosts participating in the broadcast. The logs are sent online to a log server during the event. The data is sent via TCP to avoid interfering with the overlay traffic, and the rate is limited to 20Kbps.

A reliable logging infrastructure is important for post-mortem analysis. Since the protocol runs in a distributed fashion, it is difficult to recreate the protocol state with missing data from subset of hosts. We chose to write our own logging program instead of leveraging the popular File Transfer Program (FTP) for two reasons. (i) FTP is designed to transfer files but not a stream of text. We tried to save the log data locally as a file and transfer the file when the host leaves the broadcast. However, we found a low transfer success rate, likely because users kill dislike lingering open process when they leave the broadcast event, and would actively terminate the transfer program. (ii) We observe that the TCP connections are frequently broken. We need an external mechanism to restart the connection, and transmit only the data that has yet received by the log server. Retransmitting the data from the start can waste valuable upstream bandwidth for hosts like DSL. This is difficult to achieve without modifying the FTP program.

We also monitor the broadcast system in real-time. The broadcast system involves many com-

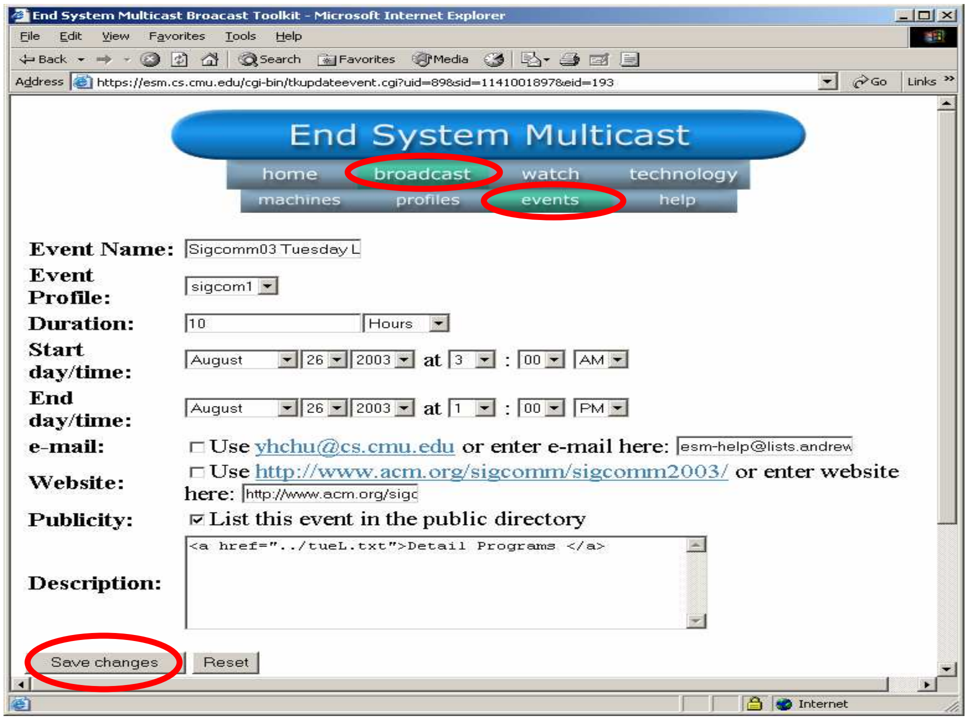

Figure 3.6: A Web interface for publishers to configure Internet broadcasting. A publisher needs to specify the source and encoding machines, the audio/video rates and encoding, and the event descriptions.

ponents running on different machines. Failure of any critical component can handicap the entire system. Through a set of monitoring tools we provide, a publisher can quickly diagnose the problems and repair faulty components (i.e. reboot a crashed machine, or reconnect a network wire). Some monitoring tools even automatically restart software components when they fail (through OS-level process monitoring). Specifically, the publisher software, the Web server, and the log server are monitored and restarted automatically if the processes die unexpected. The monitor statistics are accessible via a friendly interface through the Web portal, which is describe further in Section 3.2.7.

3.2.7 Web Portal and User Interface

Setting up a video broadcast system is quite complex. It involves proper configurations of many software components. The system can stop functioning if any one component misconfigures or fails.

To attract as many publishers and viewers to use our system, the system must be easy to set up and use. Our first iteration of the system took 5 graduate students 2 days to manually configure the machines, which was a serious barrier for deployment. We have since designed Web interfaces to accept event-specific customization, and automated machine configuration with scripts. With these improvements, the first setup time is typically less than 30 minutes for publishers and less than 5 minutes for viewers. Subsequent invocation can be done within a minute. Below we walk through an example of a setup process from the publisher and the viewer perspective.

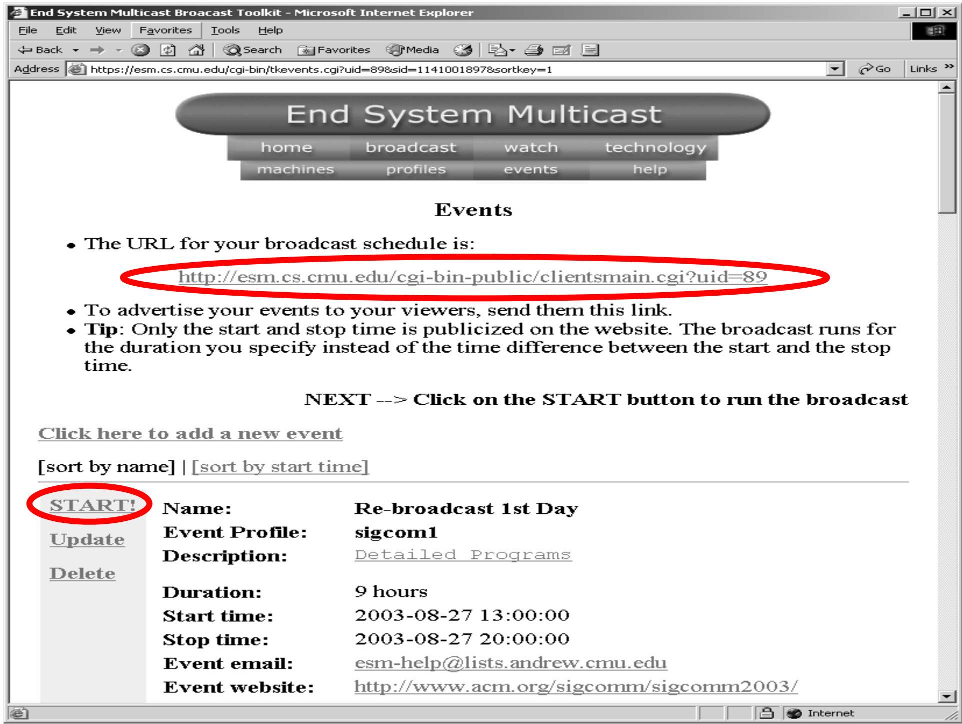

Figure 3.7: A Web interface for publishers to manage Internet broadcasting. a publisher can start, stop, and monitor the broadcast in progress.

- *Publisher Registers Event*: The publisher first enters the necessary information about the machines used (encoder/source IP address/port), the video format (codec, data rate), and the broadcast event on the Web interface. Figure 3.6 depicts an example where CMU (the publisher) announces a live Internet broadcast about its 2004 commencement. It includes a start time, duration, and the text descriptions about the event. The publisher can choose whether the event is public. If the event is public, anyone can find the event in a public directory at our portal and tune in.

- *Publisher Broadcasts Event*: Once the information is configured, the publisher can start broadcasting the event with a click on the button, as shown in Figure 3.7. The Web portal also assigns an URL, which uniquely identifies the broadcast event. The publisher can refer this event to the interested viewers with this URL. If the event is private, only those who know the URL can tune in the broadcast.

- *Viewer Searches Event)*: When viewers come to our Web portal (esm.cs.cmu.edu), they can find a list of upcoming public broadcast events. An example is show in Figure 3.8, which announces the CMU commencement broadcast. Right now the list is short, and so viewers can easily find the event from the list sorted by time and name. If in the future the list contains many registered events, then the Web portal should provide a searching capability.

- Viewer Joins Event: Once a viewer clicks on the event URL, there are two simple steps to join the broadcast (Figure 3.9). First, it must install the media player and our End System

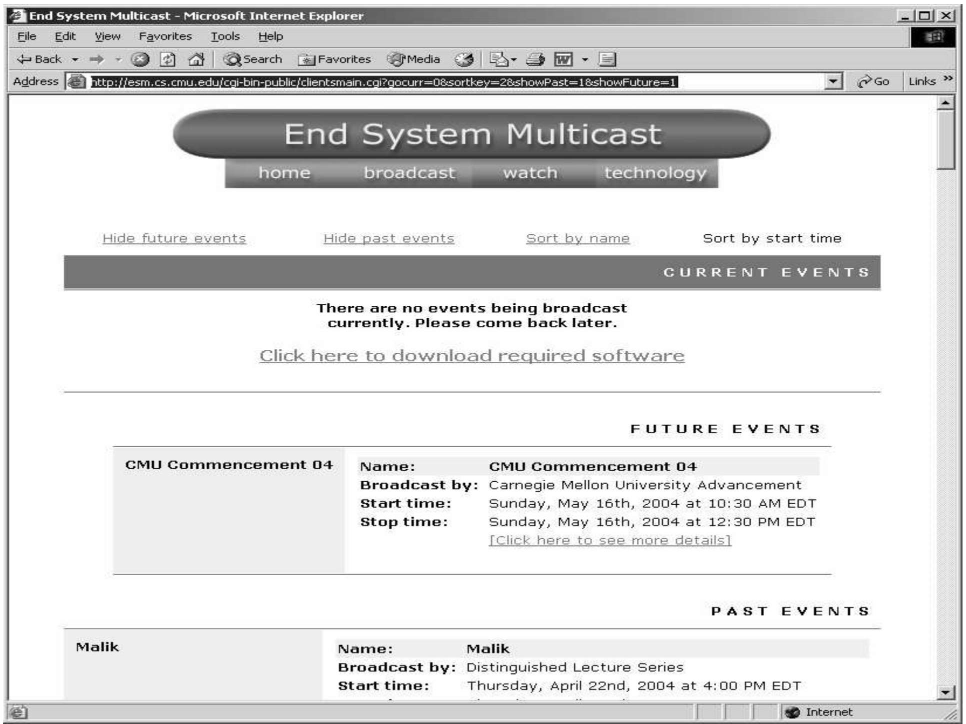

Figure 3.8: A listing of broadcast events. The listing allows viewers to find broadcast events of their interests.

Multicast software on her computer. The software is universal for all broadcast events and the viewer needs to install it just once. Next, she clicks on a link, which automatically invokes the media player and the protocol software to join the corresponding event. Most users takes a few minute to install our software and a few seconds to join the broadcast.

During an event the publisher has online performance statistics about the viewers in graphical form. This includes group size over time, performance of individual hosts (bandwidth and loss), and the tree structure produced by the protocol. This monitoring feature is helpful in providing immediate feedback to the publishers when there are problems with the broadcast.

3.3 Deployment Experience

Unlike a research prototype, a system in real deployment must be reliable. The publishers have a strong desire for the Internet broadcast to succeed, particularly when the event is heavily advertised. Section 3.3.1 describes our methodology in testing the broadcast system. Section 3.3.2 describes the scope of the system deployment.

3.3.1 Testing

To build a robust code, we have used several testing environments. These environments test different aspects of the system as described below. The first two environments, local area testbed and

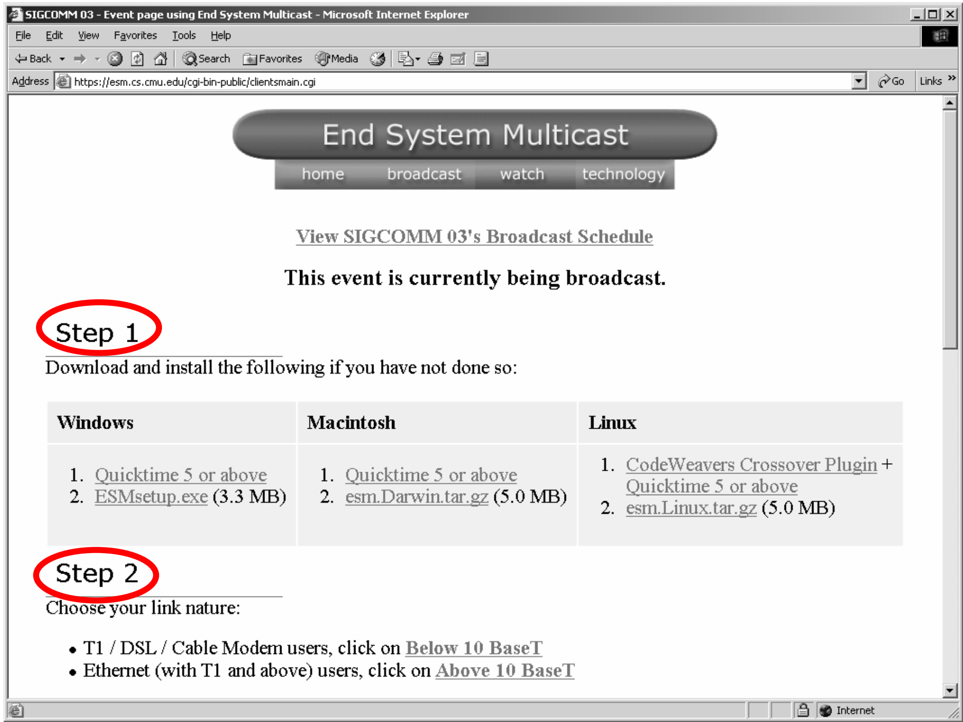

Figure 3.9: A Web interface for viewers to join a broadcast. There are two steps to join the broadcast: (i) install the software, and (ii) invoke the software by clicking on a Web link.

PlanetLab testbed, focus mainly on testing the scalability and heterogeneity of the overlay multicast protocol. The last three test the entire systems with respect to compatibility, and human perceived experience under heterogeneous operating environments.

- *Local Area Testbed*: We test the scalability of the overlay protocol on local area testbeds. We have used two testbeds: our lab with 100-160 machines, and Emulab [81] with 150 machines. For each machine, we run multiple instantiations of the protocol software. By sending data at low bitrate, we were able to simulate up to 1000 simultaneous instantiations in the system with various join and leave patterns. However, such an environment is homogeneous, as the packet latency and loss is very low.

- *PlanetLab Testbed*: PlanetLab [58] has a set of 100+ machines scattered on the Internet with a more diverse and realistic network characteristics in bandwidth, latency, and loss. This helps to test the performance of the protocol under a more realistic Internet environment.

- *Dummynet*: We use Dummynet [64] to test system performance under specific and reproducible network conditions. Dummynet is a flexible tool that simulates network delay and loss by manipulating the IP queue inside the kernel. With Dummynet, it is easy to emulate a host behind DSL or with high latency and/or high loss. This allows us to test special conditions where it is difficult to reproduce in local testbed or PlanetLab testbed.

	SIGCOMM 2002 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 3.1: Host distributions for two broadcast events, excluding waypoints, shown only for a portion of the broadcast.

- *Human Testing*: Sometime there is a gap between the performance information recorded in the trace and the video quality perceived by users. Thus it is important to evaluate the system visually as an end user. For example, we have found issues with lip synchronization (synchronization between audio and video streams) which can be annoying to users. This kind of issue is often overlooked if we look at trace data alone.

- *Compatibility Testing*: We face many system compatibility issues in the early stage of system deployment. The incompatibility mostly comes from a variation in OS platforms and Internet browsers. We systematically address the compatibility issues by (i) maintaining a suite of machines with all versions of Windows OS (from 98 to XP), several variants of Linux, and Macintosh OS X. (ii) For major code updates, we test each platform with various versions of Internet browsers, including Netscape and Internet explorer.

3.3.2 Scope of Deployment

Since the first public broadcast in August 2002, the system has been used by 4 content publishers and ourselves to broadcast more than 20 real events, the majority of which are conferences and lectures, accumulating 220 operational hours. In all, the system has been used by over 4000 participants. We summarize some of our key experience with regard to how successful we were in attracting publishers and viewers to use the system, the extent of our deployment, and some of the factors that affected our deployment.

- *Attracting content publishers:* One of the key challenges we face is finding content. It has been difficult to access popular content such as movies and entertainment, as they are not freely available and often have copyright limitations. However, we have been more successful at attracting owners of technical content, such as conferences, workshops and lectures. Typically event organizers have expressed considerable interest in the use of our system. However given the wariness toward adopting new technology, convincing an event organizer to use the system involves significant time and ground-work. The key element of our success has been finding enthusiastic champions among conference organizers who could convince their more skeptical colleagues that it is worth their while to try the new technology even when they are already overwhelmed by all the other tasks that organizing a conference involves. We have also learned that the video production process is important, both in terms of cutting costs given that conferences operate with low-budgets, and in terms of dealing with poor Internet connectivity from the conference sites to the outside world.

- *Viewer Participation:* Table 3.2 lists the major broadcasts, duration, number of unique participants, and the peak group size. The broadcast events attracted from 15 to 1600 unique participants throughout the duration and peaked at about 10 to 280 simultaneous participants. Most of the audience tuned in because they were interested in the content, but could not attend the events in person. The Slashdot broadcast is different in that wanting to explore a larger scale and wider audience, we asked readers of Slashdot [71], a Web-based discussion forum, to experiment with our system. While some of the audience tuned in for the content, others tuned in because they were curious about the system.

  While our deployment has been successful at attracting thousands of users, the peak group sizes in our broadcasts have been relatively low with the largest broadcast having a peak size of about 280. One possible explanation for this is that the technical content in these broadcasts fundamentally does not draw large peak group sizes. Another possibility is that users do not have sufficient interest in tuning in to live events, and prefer to view video archives. Our ongoing efforts to draw larger audience sizes include contacting non-technical organizations, and incorporating interactive features such as questions from the audience to the speaker.

- *Diversity of Deployment:* The diversity of hosts that took part in two of the large broadcasts (SIGCOMM 2002 and Slashdot), excluding waypoints, can be seen from Table 3.1. 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. We believe this demonstrates some of the enormous deployment potential of overlay multicast architectures - in contrast, the usage of the MBone [10] was primarily restricted to researchers in academic institutions.

- *Use of Waypoints:* Right from the early stages of our work on Overlay Multicast, we have been debating the architectural model for deploying Overlay Multicast. On the one hand, we have been excited by the deployment potential of an *application end-point architecture* that do not involve any infrastructure support and rely entirely on hosts taking part in the broadcast. On the other hand, we have been concerned about the feasibility of these architectures, given that they depend on the ability of participating hosts to support other children. When it came to actual deployment, we were not in a position to risk the success of a real event (and consequently our credibility and the content provider's credibility) by betting on such an architecture. Thus, in addition to real participants, we employed PlanetLab [58] machines, which we call waypoints, to also join the broadcast (also listed in Table 3.2).

  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 3.10, which plots a snapshot of the overlay during the *Conference* broadcast. The shape and color of each node represents the geographical location of the host as indicated by the legend. Nodes with a dark outer circle represent waypoints. There are two points to note. First, the tree achieves reasonable clustering, and nodes around the same geographical location are clustered together. Second, we see that

Event	Duration (hours)	Unique Hosts/ Waypoints	Peak Size/ Waypoints
SIGCOMM 2002	25	338/16	83/16
SIGCOMM 2003	72	705/61	101/61
DISC 2003	16	30/10	20/10
SOSP 2003	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 3.2: Summary of major broadcasts using the system. The first 4 events are names of technical conferences.

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, we can arrive at important implications for these architectures leading to reduced use of waypoints in subsequent broadcasts, as we have done in Section 3.6.

3.4 Analysis Methodology

We conduct off-line analysis on the performance logs collected from hosts participating in the broadcasts. Our evaluation and analysis focus on the following questions:

- How well does the system perform in terms of giving good performance to the user?
- What kind of environments do we see in practice? How does the environment affect system performance? Are there quantitative indices we can use to capture environment information?
- Using trace-based simulations on the data, can we ask "what-if" questions and analyze design alternatives that could have led to better performance?

The data that we use for the analysis is obtained from performance logs collected from hosts participating in the broadcast. We have instrumented our system with measurement code that logs application throughput sampled at 1 second intervals, and application loss rate sampled at 5 second intervals. Note that the sample period is longer for loss rates because we found from experience that it is difficult to get robust loss measurements for shorter sampling periods.

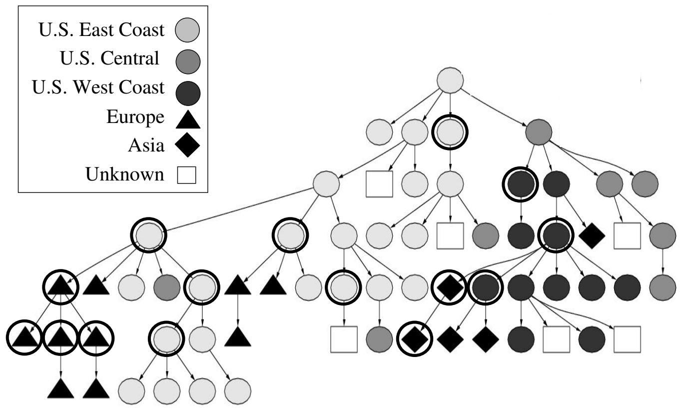

Figure 3.10: Snapshot of the overlay tree during Conference 1. Participants, marked by geographical regions, were fairly clustered. Waypoints, marked by outer circles, took on many positions throughout the tree.

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*. The following sections, report analysis on incarnations unless otherwise stated.

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.

In this section, we provide an overview of our analysis methodology. We present results from broadcasts in Section 3.5. Finally, in Section 3.6, we quantitatively analyze the performance benefits that may accrue from key design modifications motivated by our experience.

3.4.1 User Performance Metrics

We evaluate the performance that individual users observe by measuring their average and transient network-level performance. In addition, user-level feedback is also presented to provide a more complete picture of the user experience.

- *Average performance* is measured as the mean application-level throughput received at each incarnation. This provides a sense of the overall session performance.

- *Transient performance* is measured using the application-level losses that users experience. Using the sampled loss rate from the performance logs, we mark a sample 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) fraction of session for which the incarnation sees loss; (ii) mean interrupt duration; and (iii) interrupt frequency.

  Fraction of session for which the incarnation sees 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 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. This metric is biased by incarnations that have short session durations. 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.

- *User Feedback* complements the network-level metrics described above. We encouraged users to fill in a feedback form and 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, however, subjective and should be considered in conjunction with the more objective network-level metrics.

3.4.2 Environmental Factors

A self-organizing protocol needs to deal with events such as an ancestor leaving, or congestion on upstream overlay links by making parent changes. Two key factors that affect performance then are: (i) the dynamicity of the environment; and (ii) the availability of resources (parents) in the environment. The more dynamic an environment, the more frequently a host is triggered to react; the poorer the resources, the longer it could potentially take to discover a good parent.

3.4.2.1 Dynamics

The two key aspects of dynamics are: (i) group dynamics; and (ii) dynamics in the network. We measure group dynamics using mean interarrival time and session duration. We note however that the membership dynamics and overlay performance may not follow a strict cause and effect relationship. For example, users that see poor performance may leave, thus creating more dynamics in the system.

Our measurements are not conducive to summarizing network dynamics in terms of frequency and duration because of several reasons. First, we have measurements only for the subset of overlay links chosen and used by the protocol for data transfer. Second, the measurements could be biased by the protocol's behavior. For example, the observation of congestion duration may be shorter than in reality because the protocol attempts to move away from congestion and stops sampling that path. Instead, we characterize network dynamics by looking at the causes and location. The details are described in [16].

3.4.2.2 Environment Resources

Two key factors capture the resources in an environment: (i) outgoing bandwidth of hosts, which directly bounds the number of children hosts can take; and (ii) the presence of NATs and firewalls

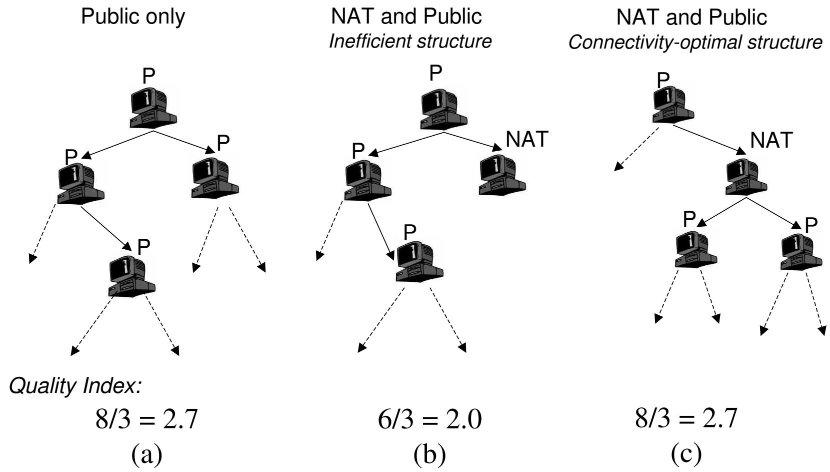

Figure 3.11: Example of Resource Index computation.

which places connectivity restrictions on parent-child relationships. In this section, we introduce a metric called the *Resource Index* to capture the outgoing bandwidth of hosts, and then extend it to consider NATs and firewalls.

We define the *Resource Index* as the ratio of the number of receivers that the members in the group could *potentially sustain* to the number of receivers in the group for a particular source rate. By number of hosts that can be potentially sustained, we mean the sum of the existing hosts in the system and the number of free slots in the system. For example, consider Figure 3.11(a), where each host has enough outgoing bandwidth to sustain 2 children. The number of free slots is 5, and the *Resource Index* is $(5+3)/3 = 8/3$. Further, for a given set of hosts and out-going bandwidth, the *Resource Index* is the same for any overlay tree constructed using these hosts. A *Resource Index* of 1 indicates that the system is saturated, and a ratio less than 1 indicates that not all the participating 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. Note that the Resource Index is sensitive to the estimation of number of slots in the system.

We have extended the definition of *Resource Index* to incorporate the connectivity constraints of NATs and firewalls, by only considering free slots available for NAT hosts. For example, in Figure 3.11(b), the number of slots available for NAT hosts is 3, and the *Resource Index* is $6/3$. However, we note that the *Resource Index* not only depends on the set of hosts, but also becomes sensitive to the structure of the overlay for that set of hosts. Thus, while Figure 3.11(c) has the same set of hosts as Figure 3.11(b), we find the number of free slots for NATs is 5 and the *Resource Index* is $8/3$.

We observe that the optimal structure for accommodating NATs is one where public hosts preferentially choose NATs as parents, leaving more free slots at public hosts which NATs can then choose as parents. Based on this observation, the *optimal* Resource Index for a set of hosts involving NATs and firewalls is defined as S/N, where $S = S_{public} + Min(S_{nat}, N_{public})$. Here, S_{public} and S_{nat} are the maximum number of children that can be supported by the public and NAT hosts, N_{public} is the number of receivers that are public hosts and N is the total number of receivers. Figure 3.11(c) is an optimal structure for the set of hosts, and it can be verified that the formula confirms to

Event	Duration (hours)	Incarnations Excluding Waypoints	Mean Session Interarrival Time (sec)	Incarnation Session Duration (min)		Entity Session Duration (min)		% Eligible Parents	
				Mean	Median	Mean	Median	All	Public
SIGCOMM 2002	8	375	83	61	11	161	93	57%	57%
SIGCOMM 2003	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 3.3: Summary of group membership dynamics and composition for the 6 larger broadcasts using the system.

the result stated above.

We wish to close with two practical issues that must be borne in mind with the *Resource Index*. First, it captures only the availability of resources in the environment, but does not account for factors such as performance of Internet paths. Second, the *Resource 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.

3.5 Analysis Results

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 SIGCOMM 2002 and 2003, we analyzed logs from one day in the broadcast. For Slashdot, we present analysis results for the first 8 hours. In this section, we will present environment characterizations and performance results of the broadcasts. The analysis will indicate strong similarities in the environment for the conference/lecture-type broadcasts. However, they differ significantly from Slashdot. When we wish to illustrate a more detailed point, we use data from the *SIGCOMM 2002* and *Slashdot* broadcasts. The *SIGCOMM 2002* broadcast is one of the largest conference/lecture-type broadcasts, and is representative of these broadcasts in terms of application performance and resources.

3.5.1 Environment Dynamics

Table 3.3 lists the mean session interarrival time in seconds for the 6 broadcasts in the fourth column. 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. Slashdot has the highest rate of group dynamics compared to all other broadcasts using our system. Note that the session interarrival times fit an exponential distribution.

Two different measures of session duration are listed in Table 3.3: 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 SIGCOMM 2002 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.

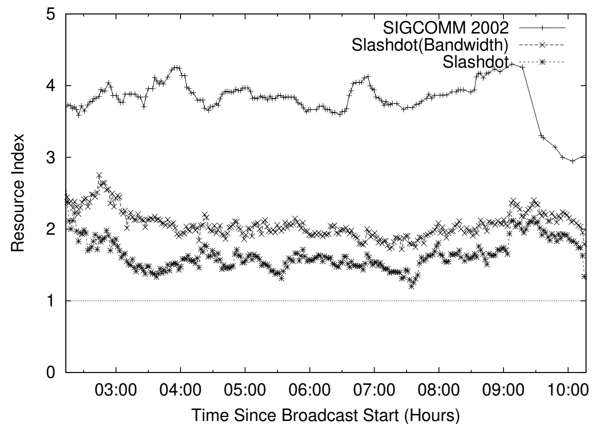

Figure 3.12: Resource Index as a function of time for (i) SIGCOMM 2002, (ii) Slashdot with bandwidth constraint, (iii) Slashdot with bandwidth and connectivity constraints.

The incarnation session duration also follows a similar trend with shorter durations. Note that SIGCOMM 2003 and Lecture 1 have very short median incarnation session durations. This is caused by 1 or 2 entities testing the system, joining and leaving frequently. Once we removed such entities, the median went up to 12 minutes or more, bringing it closer to the other 3 conferences and lectures.

3.5.2 Environment Resources

We look at the percentage of incarnations in the system that were eligible as parents, the last 2 columns in Table 3.3. 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. Note that we did not have NAT/firewall support in the SIGCOMM 2002 broadcast.

Figure 3.12 depicts the Resource Index of the system as a function of time of the broadcast. The top and the lowest curves represent the *Resource Index* for the *SIGCOMM 2002* and *Slashdot* broadcasts, and are consistent with the definition in § 3.4.2.2. We note that the lowest curve corresponds to the actual overlay tree that was constructed during the broadcast. The middle curve, *Slashdot (Bandwidth)* considers a hypothetical scenario without connectivity constraints (that is, all NAT/firewall hosts are treated as public hosts). The SIGCOMM 2002 broadcast has a Resource Index of 4, potentially enough to support 4 times the number of members. In contrast, the *Slashdot (Bandwidth)* has a Resource Index of 2, and *Slashdot* has a Resource Index that is barely over 1. Thus, not only was the distribution of out-going bandwidth less favorable in the *Slashdot* broadcast, but also the presence of connectivity constraints made it a much harsher environment.

3.5.3 Performance Results

The previous analysis indicates that 5 of our broadcasts have similar resource distributions and dynamics patterns, but the Slashdot environment was more diverse and more dynamic. This section

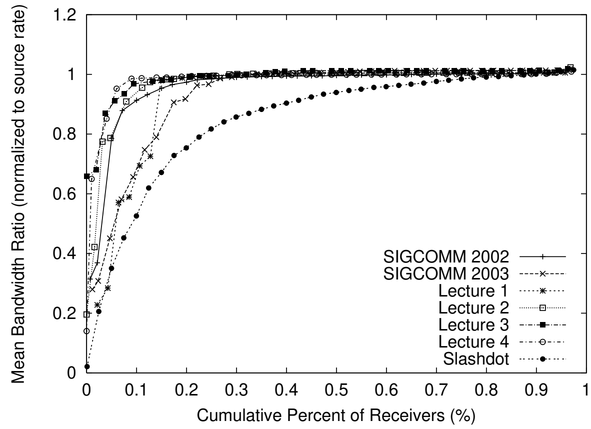

Figure 3.13: Cumulative distribution of mean session bandwidth (normalized to the source rate) for the 6 larger broadcasts.

	Setup ease	Audio Quality	Video Quality
SIGCOMM 2002	95%	92%	81%
Slashdot	96%	71%	66%

Table 3.4: Summary of user feedback for two broadcast events. Each number indicates the percentage of users who are satisfied in the given category.

evaluates how the system performs.

Figure 3.13 plots the cumulative distribution of mean session bandwidth, normalized to the source rate for the 6 broadcasts. Five of the broadcasts are seeing good performance with more than 90% of hosts getting more than 90% of the full source rate in the SIGCOMM 2002, Lecture 2, and Lecture 3 broadcasts, and more than 80% of hosts getting more than 90% of the full source rate in the SIGCOMM 2003 and Lecture 1 broadcasts. In the Slashdot broadcast, fewer hosts, 60%, are getting the same performance of 90% of the full source rate.

To better understand the transient performance, and performance of different stream qualities, we zoom in on the *SIGCOMM 2002*, which we will refer to as *Conference*, and *Slashdot* broadcasts. Figure 3.14 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 Conference broadcast, for incarnations that stay for at least 1 minute. For the Conference 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 analyze the performance of the hosts that are seeing the worst performance, and find that these are mostly hosts that are fundamentally constrained by their access bandwidth. 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

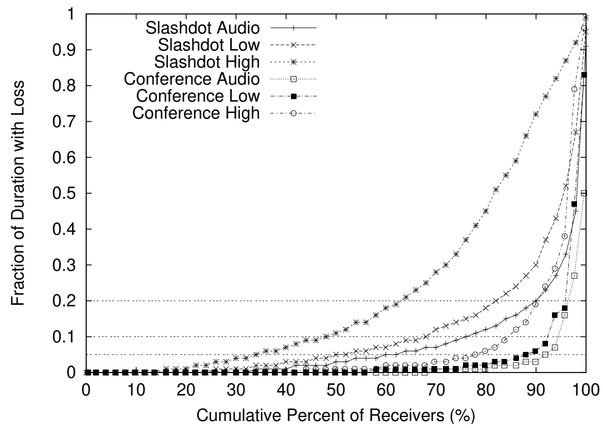

Figure 3.14: Cumulative distribution of fraction of session time with more than 5% packet loss of hosts in the two broadcasts.

audio and low quality streams are seeing better performance than the high quality because of the use of the priority buffer described in Section 3.2.2. For sessions with a high loss rate of high quality video, the low quality one was actually displayed to the user.

Next, we analyzed the interrupt duration and found that the interrupt duration is typically short for all 3 streams in Conference, and low quality video and audio in Slashdot. More than 70% 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. Roughly 60% of hosts see a mean interrupt duration of longer than 10 seconds.

We have also analyzed the cumulative distribution of the frequency of interrupts seen by each incarnation. We find that the interrupt frequency is higher for Slashdot, probably reflecting the more dynamic environment. For example, in the Conference 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.

User Feedback: Table 3.4 summarizes statistics from a feedback form users were encouraged to fill when they left the broadcast. Approximately 18% of users responded and provided feedback. Most users were satisfied with the overall performance of the system, and more satisfied with the overall performance in the Conference broadcast, which is consistent with the network level metrics in Figures 3.13 and 3.14.

3.6 Lessons Learned

Our experience over the last year, substantiated with data and analysis, has pointed us toward four key design lessons that are guiding future refinements of our system.

Our first lesson sheds light on the potential of *purely application end-point based* overlay multicast architectures that rely entirely on the hosts taking part in the broadcast. As discussed in Section 3.3, our deployment used waypoints, additional hosts that help increase the resources in the

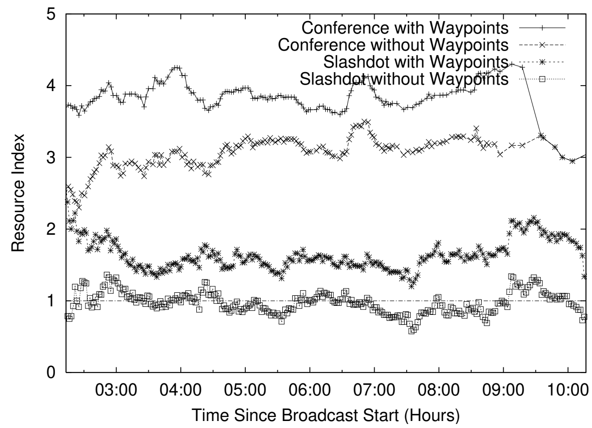

Figure 3.15: Resource Index as a function of time with and without waypoint support.

system but were otherwise no different than normal clients. We analyze how important the resources provided by waypoints was to the success of our broadcasts.

Our next three lessons deal with techniques that can enable good performance in environments with low resource index, even in the absence of waypoints. The analysis for these lessons assume that the resources provided by waypoints is unavailable, and consequently a purely application end-point architecture.

Lesson 1: There is opportunity to reduce the dependence on waypoints and use them in an on-demand fashion.

In order to understand whether or not waypoints are necessary to the success of a broadcast, we look at Figure 3.15 which plots the *Resource Index* in the Conference and Slashdot broadcasts, with and without waypoints. The Conference broadcast had enough capacity to sustain all hosts even without waypoint support. Furthermore, most of the broadcasts, similar to the Conference broadcast, are sustainable using a purely application end-point architecture. 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.

On the other hand, we find that the connectivity constraints in the Slashdot broadcast resulted in a low *Resource Index* that occasionally dipped below 1 in Figure 3.15. This indicates that it was not feasible to construct an overlay among all participating hosts that could sustain the source rate. Dealing with such environments can take on two complementary approaches (i) design 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 lessons in this section), or (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. For ease of deployment, our objective is to explore both approaches and gradually decrease the dependence on waypoints, using them as a back-up mechanism, only when needed.

We note that 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 and

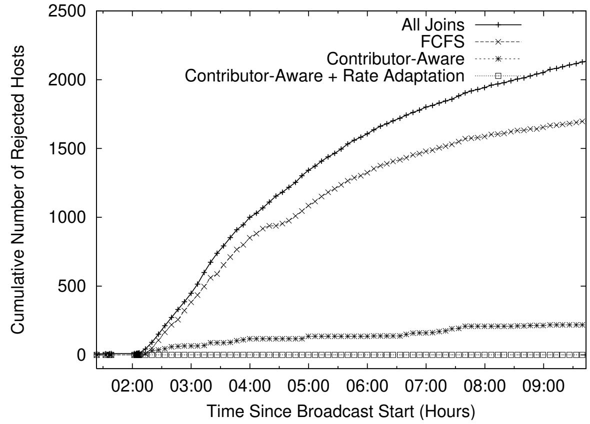

Figure 3.16: Number of rejected hosts under three different protocol scenarios in the simulated Slashdot environment.

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.

Lesson 2: Exploiting heterogeneity in node capabilities through differential treatment is critical to improve the performance of the system in environments with low Resource Index. Further, there is considerable benefit to coupling such mechanisms with application-specific knowledge.

If the Resource Index dips below 1, 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 3.16 shows the performance of the policies.

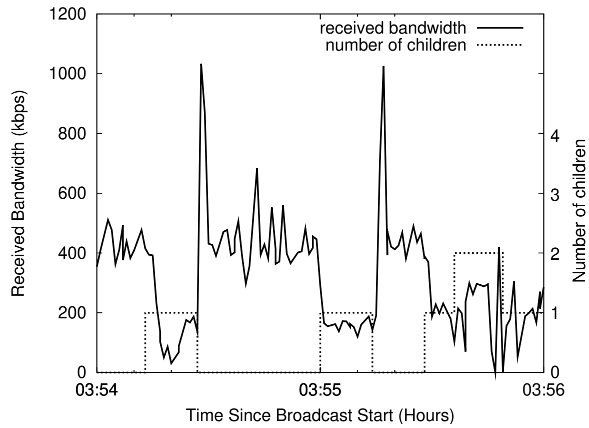

Figure 3.17: An example of a misconfigured DSL host taking children, causing poor performance to itself and its children.

	10+Mbps	Below 10Mbps	Total
User truthful	11.1%	60.8%	71.9%
User lied	5.4%	4.9%	10.3%
User inconsistent	4.3%	13.5%	17.8%
Total	20.8%	79.2%	100.0%

Table 3.5: Accuracy in determining access bandwidth based on user input in Slashdot.

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 Resource for some portion of the session.

Our results demonstrate the theoretical 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 (see next lesson), 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.

Lesson 3: Although many users are honest about contributing resources, techniques are needed for automatically estimating the outgoing access bandwidth of nodes.

As the previous lesson indicates, it is important to design protocol techniques that differentially treat nodes based on their contributions. An issue then is determining the contribution level of a node to the system, and in particular, determining the outgoing access bandwidth of a node. In our current system, the user is asked if his access bandwidth has a 10Mbps up-link to the Internet to help determine whether the host should have children (Section 3.2.1). This approach is susceptible to free-loaders[66], where a user declares that he has less resources than he really does. However,

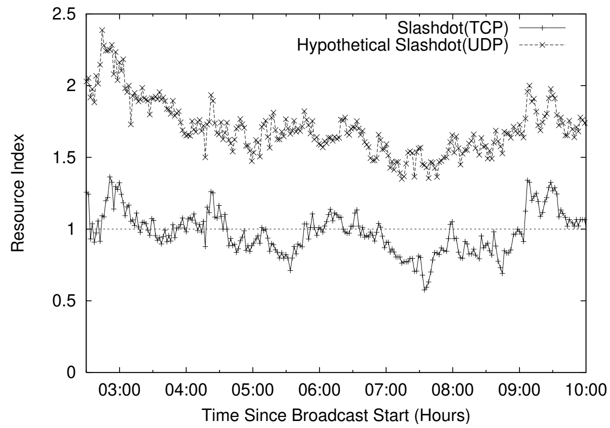

Figure 3.18: Resource Index comparison of two connectivity solutions for NAT/firewall: (i) Slashdot (TCP), (ii) Hypothetical Slashdot (UDP).

an equally damaging problem in the context of Overlay Multicast is when a user declares he has more resources than he does. To see this, consider Figure 3.17 which depicts the performance of a DSL host that lied about having a 10Mbps up-link to the Internet, during the *Slashdot* broadcast. Whenever the host accepts a child, it affects not only the child's performance, but also its own performance. Further, a similar problem arises when a host can support less children (e.g. 4) than it claimed (e.g. 6). In a future design that prioritizes hosts that contribute more (Lesson 2), these effects can get further exacerbated.

To appreciate how reliable users were in selecting the correct access bandwidth in the Slashdot broadcast, consider Table 3.5. Each column represents a true access bandwidth, and each row represents a particular type of user behavior. "User Inconsistent" refers to users that had joined the group multiple times during the broadcast, and had selected both 10+Mbps option and lower than 10 Mbps option between consecutive joins, perhaps trying to figure out whether the choice yielded any difference in video quality. We determined the real access bandwidth using an off-line log analysis involving the following techniques: (i) DNS name, (ii) the TCP bandwidth of the upload log, (iii) online bottleneck bandwidth measurement, and (iv) Nettimer [41] from our university to target hosts. Since no single methodology is 100% accurate, we correlate results from all these techniques. We omit the details for lack of space.

From the table, we see that overall 71.9% of hosts are truthful. However, for the 20.8% of hosts that were behind $10Mbps$ links, only half of them (11.1% of total) were truthful. Our trace-based simulation on the Slashdot log indicates that on average, this results in a 20% increase in *Resource Index* . Further, we find that while 79.2% of the users were behind links lower than $10Mbps$, about 4.9% chose the higher option or were being inconsistent (13.5%) about their connectivity.

We have been experimenting with techniques to explicitly measure the static outgoing access capacity of hosts and passively monitor the performance of parents to dynamically track their available bandwidth. These techniques show promise and we hope to deploy them in the future.

Lesson 4: Addressing the connectivity constraints posed by NATs and Firewalls may require using *explicit NAT/firewall-aware heuristics in the protocol.*

In light of our experience, NATs and firewalls can constitute an overwhelming fraction of a broadcast (for example, 50%-70% in *Slashdot*), and thus significantly lower the *Resource Index*. Clearly, using UDP as the transport protocol could improve the situation by increasing the amount of pair-wise connectivity, particularly connectivity between Full-Cone NATs. However, a less obvious improvement, which we briefly presented in Section 3.2.5 is to make the self-organizing protocol explicitly aware of NAT/firewalls. In particular public hosts should preferentially choose NATs as parents, leaving more resources available for NATs/firewalls.

We now evaluate the potential of these two design improvements to help determine whether or not the additional complexity is worth the performance gains. Figure 3.18 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 the optimal Resource Index that can be achieved with a TCP-based protocol. The topmost curve corresponds to the optimal Resource Index with UDP and a NAT/firewall-aware self-organizing protocol. We see a significant increase of 74%. The combination of the two techniques above can significantly improve the *Resource Index*. Both techniques are being implemented in the latest version of our system and will soon be used for upcoming broadcasts.

3.7 Related Works

In this section, we discuss how our work relates to (i) other existing Internet broadcast systems and (ii) work in the Overlay Multicast community.

Broadcast Systems: The MBone [10] Project, and its associated applications such as vic [46], vat [34], and MASH [45] made a substantial 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.

Commercial entities, such as Akamai [2] and Real Broadcast Network [61], 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.

Recently, several peer-to-peer broadcast systems have been built by commercial entities [3, 13, 75] and non-profit organizations[55]. 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.

Overlay Multicast: Since overlay multicast was first proposed four years ago many efforts [29, 36, 14, 43, 60, 85, 12, 4, 53, 79, 39, 11] have advanced our knowledge on protocol construction by improving performance and scalability. Most of this work has been *protocol-centric*, and has primarily involved evaluation in simulation, and Internet testbeds such as PlanetLab. In contrast, this work adopts an *application-centric* approach, which leverages experience from actual deployment to guide the research. We address a wide range of issues such as support for heterogeneous receivers, and NATs and firewalls, which are not typically considered in protocol design studies. To our knowledge this work is among the first reports on experience with a real application deployment based on overlay multicast involving real users watching live content. We believe our efforts complements ongoing research in overlay multicast, by validation through real deployment, and providing unique data, traces and insight that can guide future research.

The overlay protocol that we use is distributed, self-organizing and performance-aware. We use a distributed protocol, as opposed to a centralized protocol [56, 53], to minimize the overhead at the source. The self-organizing protocol constructs an overlay tree amongst participating hosts in a tree-first manner, similar to other protocols [36, 79, 29], motivated by the needs of single source applications. In contrast there are protocols that construct a richer mesh structure first and then construct a tree on top [14], or construct DHT-based meshes using logical IDs and employ a routing algorithm to construct a tree in the second phase [12]. Such protocols are typically designed for multi-source or multi-group applications.

In our protocol, members maintain information about hosts that may be uncorrelated to the tree, in addition to path information, while in protocols like Overcast [36] and NICE [4], group membership state is tightly coupled to the existing tree structure: While Yoid [29] and Scribe [12] also maintain such information, the mechanisms they adopt are different. Our system uses a gossip protocol adapted from [63], while Yoid builds a separate random control structure called the mesh, and Scribe constructs a topology based on logical identifiers.

Overcast [36] discusses adaptation to dynamic network metrics such as bandwidth. Our experience indicates that a practical deployment must consider several details such as dynamic tuning of network detection time to the resources available in the environment, consider hosts that cannot sustain the source rate, and consider VBR streams, and indicate the need for further research and understanding in this area.

Recent work such as CoopNet [53], and SplitStream [11] has demonstrated significant benefits by tightly coupling codec-specific knowledge and overlay design. In these works, the source uses a custom codec to encode the multimedia stream into many sub-streams using multiple description coding, and constructs an overlay tree to distribute each sub-stream. This approach not only increases overall resiliency of the system, but also enables support for heterogeneous hosts by having each receiver subscribe to as many layers as its capacity allows. While we believe this a great direction for future research, 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. We hope to leverage ideas from this approach as the research attains greater maturity, and when custom codecs become available.

NATs and Firewalls: Several efforts such as UPnP [76] and STUN [65] focus their efforts in enabling connectivity of NATs and firewalls. Our focus has been on the interplay between the application and NAT/firewall support. In particular, we have examined how the connectivity constraints imposed by NATs and firewalls can impact overlay performance, and on issues related to the integration of protocol design with NATs and firewalls. While Yoid [29] supports NATs and firewalls, it supports such hosts as children only, whereas we try to use NATs as parents when possible. We believe this is one of the first reports on experience with an overlay multicast system in the presence of NATs and firewalls.

3.8 Summary

In this chapter, we have reported on our operational experience with a broadcast system based on Overlay Multicast. To our knowledge this is among the first reports on experience with real application deployment based on Overlay Multicast, involving real users. Our experience has included several positives, and taught us important lessons both from an operational deployment stand-point, and from a design stand-point.

Our system is satisfying the needs of real content publishers and viewers, and demonstrating the potential of Overlay Multicast as a cost-effective alternative for enabling Internet broadcast. The system is easy to use for both publishers and viewers. We have successfully attracted over 4000 users from diverse Internet locations to use our system. However, we have had limited success in attracting larger scales of participation, primarily because of the difficulty in getting access to non-technical content. 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 with a low *Resource Index*, users see good performance in audio and low Resource video.

Getting the system deployed has frequently required finding an enthusiastic champion of the technology to convince their colleagues to use it. This has raised the stakes to ensure the success of a broadcast, which could in turn trigger further interest in the use of the system. Consequently, we have needed to use stable and well-tested code in our deployment, rather than code that implements the latest performance enhancements. Another consequence has been our use of waypoints, additional hosts that help increase the resources in the system, but were otherwise no different than normal clients. The use of waypoints has been motivated by the need to balance between conflicting goals - on the one hand we want to understand the resource availability in purely application end-point architectures; on the other hand we need to have a series of successful broadcasts in the first place before such knowledge can be obtained.

Our subsequent analysis has investigated the potential of *purely application end-point architectures*, that do not rely on the use of waypoints. Our analysis both show the promise for such architectures, but also the need to incorporate additional key design elements. For most of our broadcasts, there is sufficient bandwidth resources to enable a solution purely within the application end-point framework. In broadcasts with lower Resource Index, techniques that exploit the heterogeneity in node capabilities through differential treatment and application-specific knowledge bear significant promise. Our broadcasts have also forced us to better appreciate the connectivity constraints posed by NATs and firewalls, and have led us to investigate explicit NAT/firewall-aware heuristics in the protocol. While our lessons have been derived in the context of our system, we believe they are of broader applicability to the community as a whole.

Chapter 4

Incentive Mechanisms

One fundamental assumption about End System Multicast is that end systems (peers) are willing to contribute resource for data replication. Previous chapters implicitly assume that peers are *completely altruistic*, i.e. they contribute *all* of their resource when they join the broadcast group. With the prevalence of free-riding in peer-to-peer file-sharing applications, it is not clear whether peers would always behave with complete altruism. In most cases, peers contribute resource only if they see a clear incentive. In this chapter, we ask the following two questions:

- How to design an incentive mechanism that can effectively harness resource from peers to achieve good overall system performance?
- How to incorporate the incentive mechanism into existing overlay multicast protocols, and what is the performance overhead?

We propose a new *taxation* model [15], where resource-rich peers contribute more resource to the system, and subsidize for the resource-poor peers. This redistribution of resource significantly improve system performance compared to a *Bit-for-Bit* model. In a Bit-for-Bit model, peers contribute only as much as received, which is adopted by several existing protocols [11, 53, 20].

Taxation has one important precondition. There must exist *asymmetry of roles*, where one entity (role) is empowered to enforce tax payment on individuals according to a predefined tax schedule. We believe the P2P broadcast application is in an unique position to satisfy this precondition. In P2P broadcast, the *publisher* of the media stream is the natural empowered entity. The publisher owns the content and can therefore choose the means in which peers participate in the system (via proprietary software). In another word, the publisher can freely design a game and enforce the rules of the game. Peers participating in this game are strategic. They individually own the bandwidth resource and are strategic in minimizing the cost of contributing resource while maximizing the benefit of the video quality received.

We note that taxation provides a *direct* mapping between contribution and benefit, in contrast to other incentive mechanisms based on currencies [82, 78, 30, 21] and/or reputation [9, 40, 31], which provide *indirect* mappings between contribution and benefit. The indirection is necessary if a peer's contribution and consumption are temporally separated. However, this is not the case for P2P broadcast. Therefore, the adoption of a taxation scheme avoids the overhead and security vulnerabilities of maintaining persistent state (e.g., tokens).

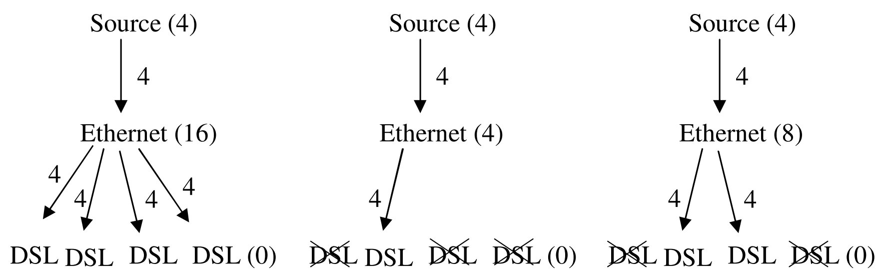

Figure 4.1: Examples to illustrate the effect of altruism on the contributing and received bandwidth of peers. Each unit bandwidth is 100Kbps.

We study the performance and implementability of standard taxation schemes from the public finance literature. We find that a linear tax schedule, with a single marginal tax rate and a demogrant, provides significant social welfare improvements over the Bit-for-Bit scheme, especially with heterogeneous populations. Furthermore, the taxation scheme is implementable. By employing techniques such as multiple description codec (MDC), priority, and preemption, the scheme works well under dynamic peer environments. Evaluations show that the taxation scheme achieves high efficiency and high compliance without incurring significant overhead [19].

The rest of the chapter is organized as follows. In Section 4.1, we argue the suboptimality of Bit-for-Bit and motivate the need for taxation. Section 4.2 identifies unique characteristics of the P2P broadcast application where taxation is enforceable. Section 4.3 presents a simple construction of the taxation scheme. This taxation scheme can be incorporated into a distributed protocol, as described in Section 4.4. We study the performance and feasibility of taxation in 4.5 using simulation with traces from real broadcast events. Finally, we present the related works in Section 4.6, discuss important limitations in Section 4.7, and summarize our contributions in Section 4.8.

4.1 Background

In this section, we argue why the Bit-for-Bit model is suboptimal in today's Internet environment and motivate the need for *subsidy*. To illustrate the importance of subsidy, we provide an example how the degree of subsidy affect peer performance with a single tree overlay structure. Our example also indicates the limitation of a single tree approach in supporting a flexible range of available subsidy. We then describe two basic building blocks proposed in [53, 11] that we leverage in this chapter: multiple tree structure and MDC.

4.1.1 Conventional Approach: Bit-for-Bit

The conventional incentive model is *Bit-for-Bit*, where peers receive only as much as they contribute. This model is attractive in its apparent fairness and has been adopted by protocol designers [11, 53, 20]. Free-riders who contribute no resource will receive no bandwidth in return. However, our deployment experience predicts that it will perform poorly in today's Internet environment.

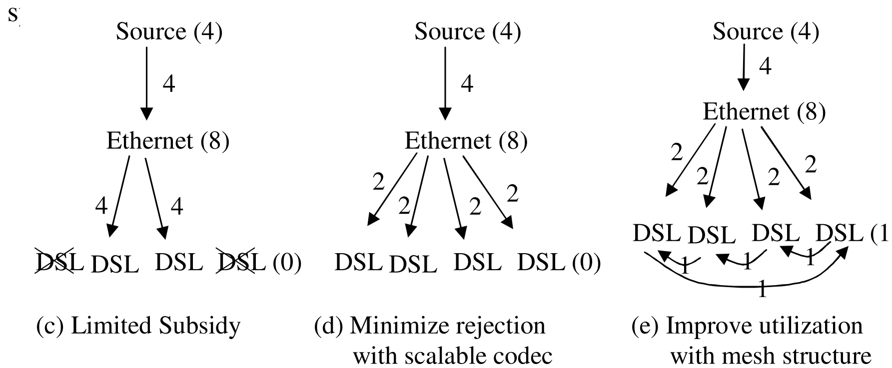

Figure 4.2: Examples to illustrate the two optimizations used in the Chapter. This is an continuation from the examples in Figure 4.1.

In a P2P environment for audio/video broadcasting, bandwidth is the bottleneck resource. In our current system, a reasonable quality motion audio/video stream takes at least 300Kbps to encode. However, many of the peers are behind DSL and cable modems. These resource-poor peers have asymmetrical bandwidth capacity with low forwarding capacity (100-200Kbps) and high receiving capacity (600-1200Kbps). In our broadcasts, up to 80% of the peers are resource-poor. If we adopted the *Bit-for-Bit* model, these resource-poor peers would only receive lower video bitrate (100-200Kbps), even though they have enough capacity to receive at a much higher bitrate. The net result is that these resource-poor peers would *not* participate in the broadcast due to the poor quality.

We illustrate the suboptimality of Bit-for-Bit with an example. Figure 4.1 depicts an example overlay multicast tree with a single source and 5 peers. The source broadcasts a video stream at a rate of 400Kbps with no special encoding (no MDC). The maximum forwarding capacity of the source is 400Kbps. The peers are behind either a DSL connection (resource-poor) or Ethernet connection (resource-rich). Peers behind DSL can receive the video stream but not forward one. The peer behind Ethernet has at least 1600 Kbps forwarding capacity. If we adopted the *Bit-for-Bit* scheme, most of the resource-poor peers would be rejected from the broadcast. The resource-rich peer contributes just 400Kbps as shown in Figure 4.1(a). With limited forwarding capacity, only one DSL can receive the video stream. The three other DSL peers are rejected from the broadcast.

To accommodate resource-poor peers, who would be otherwise not able to participate, we would like to design a system that incentivizes high-resource peers to contribute more bandwidth and *subsidize* for the low-resource peers. The amount of subsidy determines the system performance. For example, if the Ethernet peer contributes twice the source rate, the overlay can in turn accept two DSL peers, as shown in Figure 4.1(b).

The degree of subsidy of resource-rich peers has significant impact on system performance. We note that our system described in the previous chapter assume a maximum degree of subsidy, as shown in Figure 4.1(c). The resource-rich peer contributes 1600Kbps to the overlay and yields an optimal performance where all peers receive a source rate of 400Kbps.

4.1.2 Required Building Blocks

To support a flexible range of available subsidy, we leverage two techniques in the literature: *multiple tree structure* and *multiple description codec (MDC)* [53, 11].

Content publishers typically want to accept as many viewers in the broadcast overlay as possible, and it is not desirable to reject peers as shown in Figure 4.2(c). In an overlay environment where the forwarding capacity is fixed, we can reduce the number of rejection by giving some peers lower video bitrate, but the resulting video quality is still acceptable to them. For example, we can accept all the DSL peers if the Ethernet peer forwards 200Kbps video stream to each DSL, as shown in Figure 4.2(d). However, this requires a scalable video codec, such as multiple descriptions coding (MDC), that allows peers to receive different video bitrate.

Several overlay multicast protocols have used MDC to support receiver heterogeneity and improve resilience to machine failure and network congestion [53, 11, 39]. We use MDC for a different purpose. MDC gives peers the flexibility to contribute and receive small increments of bandwidth, instead of discrete increments of the source rate. Such flexibility helps us to express the altruism policy in finer granularity. For example, if each stripe of video is 50Kbps, then the Ethernet peer can contribute 450Kbps, and one DSL peer would receive an additional 50Kbps.

The overlay structure constructed by the protocol affects the efficiency in utilizing the valuable forwarding capacity. In the previous examples, the overlay structure is a tree, and the forwarding capacity at the leaf nodes cannot be used. Various protocols [39, 53, 11] build alternate overlay structures, such as mesh or multiple disjoint trees, to improve bandwidth utilization. As an example, consider a mesh structure in Figure 4.2(e). The DSL hosts can receive additional 100Kbps by forwarding non-overlapping data to each other. In the next section, we assume the overlay runs an efficient protocol that can utilize *all* forwarding capacity (subject to MDC channel granularity). In Section 4.4, we show how to incorporate policy constraints into a protocol that constructs multiple disjoint trees structure.

4.2 Model of P2P Broadcast

There are two entities in the P2P broadcast system we consider: publisher and peers. A publisher makes a live video stream available on the Internet, and peers who are interested in the stream join the P2P system. Peers constructs an overlay structure in a distributed fashion and disseminate the video stream along the overlay. Both entities have mutual incentive to use the P2P system. By delegating the task of data forwarding to the peers, publishers can avoid the costly bandwidth provisioning to support a large number of viewers. The peers have an incentive to help the publisher in exchange for enjoying the video that might otherwise be unavailable.

In this scenario, bandwidth is the valuable resource. High quality real-time video broadcasting requires the availability of high bandwidth (at least hundreds of Kbps) that is persistent over time. Thus we consider the peers' bandwidth capacity as their "wealth" for subsidy (taxation).

We model the bandwidth capacity of a peer i with two parameters: forward capacity (F_i) and receive capacity (R_i). F_i and R_i represent the upper bound bandwidth that a peer can contribute to and receive from the P2P system, respectively. We do not model congestion in the core of the network, as congestion happens mostly at the access links on the Internet today. Therefore, a peer

N	Number of receivers in the broadcast overlay
s_{max}	Maximum source rate
m	Number of channels in MDC encoding
t	Linear tax rate (between 1.0 and ∞)
G	Demogrant (between 0 and s_{max})
F_i, R_i	Forwarding and receiving capacity of host i
f_i, r_i	Bandwidth host i is commited to forward and expected to receive in return
${f_i}^*$, ${r_i}^*$	Actual bandwidth host i forwards and receives at run time
c_i, b_i, u_i	Cost, benefit and utility of host i
S	Social welfare of the overall system
${r_i}^+$	Entitled bandwidth of host i

Table 4.1: Terminology and explanation used in Chapter 4.

can forward traffic to another peer as long the capacity bounds on both ends are met. A peer may choose to contribute only a portion of its capacity. f_i denotes the bandwidth peer i is committed to forward, and r_i denotes the bandwidth peer i is expected to receive in return. Thus $F_i \geq f_i$, $R_i \geq r_i$, and $\sum f_i = \sum r_i$. Table 4.1 summarizes the terminology used throughout this chapter.

However, in a real distributed environment, the bandwidth resource is not always utilized. Moreover, the resource is not always allocated as intended. To account for this discrepancy, we denote ${f_i}^*$ and ${r_i}^*$ as the actual bandwidth host i forwards and receive at run time. We explain their characteristics as follows:

- ${f_i}^*$ is always less than or equal to f_i. When ${f_i}^* \leq f_i$, the resource said to be *under utilized*. For example, if an existing child departs, the bandwidth resource is idle until a new child arrives. ${f_i}^*$ cannot be greater than f_i because peer i accepts new child only if it does not over-commits itself.

- ${r_i}^*$ may be greater or less than r_i. When ${r_i}^* \neq r_i$, the resource is said to be *not compliant* to the allocation policy, assuming the resource is fully utilized. Thus, when the resource is under utilized, the allocation cannot be compliant because some receivers will get less bandwidth. But even if resource is fully utilized, it is possible that ${r_i}^* \geq r_i$ for some i. This is mainly an artifact of the "work-conserving" nature of the proposed taxation scheme. A parent with free forwarding resource (i.e. ${f_i}^* < f_i$) will serve bandwidth to any child requesting the resource. Thus in the transient, a peer may get more bandwidth before other peers discovers and requests that resource.

- $\sum {f_i}^*$ must be equal to $\sum {r_i}^*$, assuming there is no packet loss in the network. This is because any peer who receives one bit of data must be sent by another peer in the group.

For simplicity of the discussion, we assume a centralized planner where resource is allocated fully and in compliance in the remainder of the chapter. This implies that ${f_i}^* = f_i$ and ${r_i}^* = r_i$. We will come back to discuss this discrepancy when evaluating the performance of our distributed protocol in Section 4.5.5.

A peer gains benefit b_i when it receives bandwidth from the broadcast system, and incurs cost c_i when it contributes bandwidth to the system. When faced with a taxation scheme that specifies the contributed bandwidth f_i as a function of the received bandwidth r_i, a strategic peer will choose an optimal r_i to maximize its utility u_i, subject to $r_i \leq R_i$ and $f_i \leq F_i$.

$$u_i(r_i, f_i, F_i) = b_i(r_i) - c_i(f_i, F_i) \tag{4.1}$$

The benefit function (b_i) captures the user-perceived video quality. The function should be *concave* or S-shaped in the receive bandwidth (r_i) to capture the diminishing returns of increased video bitrate on the perceived video quality. The benefit function should be independent of a peer's forward capacity (F_i) to reflect the equal desire of viewers to watch a high quality video stream.

The cost function (c_i) captures the cost of forwarding data. The function should be concave in F_i to capture economies of scale in bandwidth but convex in f_i (or more specifically f_i/F_i) to capture the effects of link congestion. We will consider specific functional forms for user benefits and costs in Section 4.5.

Finally, we model a publisher who is interested in maximizing the social welfare (S) of the system, which is simply the summation of individual utilities of the peers. The publisher may also choose a tax schedule to maximize some other objective function, e.g., system throughput, demogrant.

$$S = \frac{\sum u_i}{N} \tag{4.2}$$

4.3 Proposed Taxation Scheme

In this section, we show how to incorporate taxation into a P2P broadcast system. We first construct a suitable taxation model for P2P broadcasting in Section 4.3.1 based on the public finance literature. The main departure from the traditional taxation is that P2P broadcasting is less tolerant to a budget deficit. In Sections 4.3.2 and 4.3.3, we propose a simple taxation schedule that we use throughout the chapter. This schedule is based on linear taxation, which is widely studied in the optimal income taxation literature. We modify the semantic to ensure a budget balance. Finally in Section 4.3.4, we show how the publisher sets the tax schedule in practice.

4.3.1 Model Taxation in P2P Broadcasting

For peer i, the taxable income is r_i and the tax payment is f_i. In other words, a tax schedule specifies how much bandwidth a peer must contribute (f units) in order to receive r unit of bandwidth. In the following we list a set of requirements for taxation in the public finance literature [77].

- *Asymmetry of roles and power*: There must exist an entity empowered to set and collect taxes from the individuals. In P2P broadcasting, the publisher is the natural entity. The publisher owns the broadcast rights of the video stream, and has control over the means (the software) in which the video is distributed. The peers are assumed to be strategic. Each peer chooses its optimal contribution level f_i, and receive a corresponding amount r_i determined by the published tax schedule, in order to maximize its utility. Peers are also assumed to satisfy the *participation constraint*, i.e., their utility from participation and adherence to the taxation scheme exceeds a reservation utility derived from not doing so (e.g., not participating in the system, or modifying the software to engage in tax evasion).

We argue it is reasonable to assume viewers cannot change the protocol behavior by circumventing the software. If the publisher finds serious and wide-spread software breach, it can force all the viewers to update a new version of the software (by being incompatible with previous versions of the software). We conjecture that a serious and wide spread software breach is usually not difficult to detect.

- *Public and fixed tax schedule*: A tax schedule should be fixed and made public such that peers can adjust their strategy to maximize their utility. The tax schedule should not change (or should change at a very large time scale) to minimize system instability due to peers reacting to the changes in tax schedule.

- *Fair*: There are two types of fairness: horizontal and vertical. Horizontal fairness requires that individuals with similar wealth should bear similar tax liability. We adopt the same requirement in P2P broadcasting. Vertical fairness requires that individuals with different wealth should bear (potentially) different tax liability. In public finance, vertical fairness is more a matter of public opinion, where taxpayers can influence the tax schedule (through voting). In P2P broadcasting, the tax schedule is determined solely by the publisher. In this dissertation, the publisher aims to maximize social welfare as described in Section 4.5.1.

- *Budget balanced*: Budget is not balanced (budget deficit) if the budget expenditure exceeds the revenue from taxation. In P2P broadcasting, this means that $\sum f_i$ (budget expenditure) must be greater or equal to $\sum r_i$ (tax revenue). This is intuitive because every byte of bandwidth received by a peer must be contributed by another peer. The requirement for budget balance is more stringent in P2P broadcasting than in public finance, because taxed money collected in public finance is a *persistent* resource but the taxed bandwidth collected in live P2P broadcasting is perishable. Thus in P2P broadcasting, it is not possible to "store" the bandwidth resource and "use" it later on.

- *Efficient*: The distribution of tax expenditure (bandwidth) typically incurs an administration cost. In P2P broadcasting, the cost is the protocol overhead in allocating bandwidth. We quantify this cost in the evaluation with the *utilization* metric.

In summary, if we model a P2P broadcast system as an economic game played between the publisher and the viewers, then the publisher has a significant role in defining the rules of the taxation game. Our design leverages this asymmetry of role to implement subsidy. However, the publisher cannot set the tax schedule arbitrarily; the tax schedule should be public, fixed, and budget balanced.

4.3.2 Linear Tax Schedule

We choose a linear tax schedule, which takes on two parameters: (i) t, *marginal tax rate*, and (ii) G, lump sum grant, also known as *demogrant*. Note that if a peer does not contribute any bandwidth ($f = 0$, or free ride), it would still receive a demogrant ($r = G$). The publishers sets only the

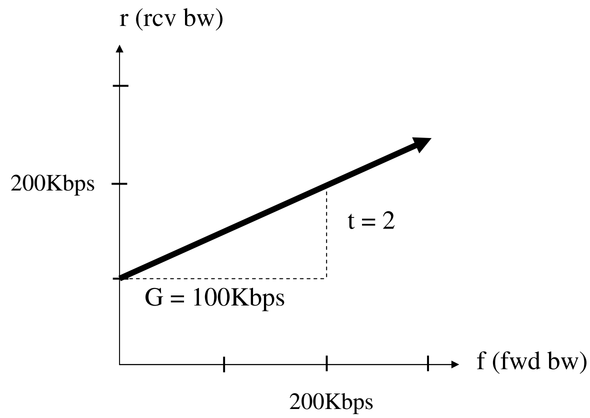

Figure 4.3: A linear tax schedule with a tax rate of 2.0 and a demogrant of 100Kbps. The received bandwidth of a peer is a function of its contributing (forwarding) bandwidth.

first parameter. The second parameter is dynamically inferred from the peer environment to achieve budget balance. This is a departure from the traditional literature in linear taxation where both parameters are configured simultaneously.

$$f = max(t * (r - G), 0) \tag{4.3}$$

The tax rate (t) must be at least 1, otherwise fundamentally the budget cannot be balanced even if the demogrant is 0. When $t = 1$, the tax schedule becomes *Bit-for-Bit* and $G = 0$. This is because when $f_i = r_i$, there is no extra tax expenditure for demogrant. When $t > 1$, the demogrant may be greater than 0. If a peer contributes more than it receives ($f_i > r_i$), the bandwidth difference goes to a *demogrant pool*. This pool of bandwidth is then evenly distributed among all peers as demogrant. An example of a linear tax schedule is shown in Figure 4.3.

Linear taxation has been widely studied in the optimal income taxation literature. Despite its simplicity over nonlinear taxes, linear taxes provide surprisingly robust results in many settings [49, 73]. [1] Our evaluation results in Section 4.5.4 are aligned with this observation. We note that there are many other taxation schedules and budget balance strategies, and defer their investigation to future work. Our goal in this chapter is to demonstrate the existence of one taxation scheme that is effective and implementable in a P2P environment.

4.3.3 Budget Balance Strategy

Given a fixed tax rate, a fixed demogrant, and a fixed peer environment, the tax budget (tax revenue minus the expenditure) can be in one of the three conditions: (i) balance, (ii) surplus, or (iii) deficit. Ideally we want the budget be balanced. Given the taxed bandwidth is a perishable resource, a surplus wastes resource (and hurt social welfare) and a deficit makes the system infeasible to operate.

Our solution is to tune the demogrant such that the budget is balanced. There are two reasons why demogrant should not be configured statically. (i) The peer environment is dynamic, and so is the composition of peers in the system. This may tip the balance of the budget (from surplus to

[1] Indeed there are many proponents for the adoption of a linear tax for U.S. federal income taxation.

deficit, for example). (ii) Even if the peer environment is relatively stable, it is difficult to set the demogrant in the first place. The publisher must know not only the bandwidth capacity of all the peers, but also their utility functions. This is private information that peers may not want to reveal to the publisher.

Deriving the demogrant value in a centralized manner takes several rounds. In each round j, the planner announces the demogrant value (G_j) to all peers. Each peer determines the best (utility-maximizing) strategy and tells the planner its f_i. The planner then finds the highest demogrant (G_{j+1}) without causing a budget deficit. The algorithm stops when $G_{j+1} = G_j$. We observe G sometimes oscillates between two values. In this case, we damp G_{j+1} to increment by a small amount. In Section 4.4, we show how to derive demogrant dynamically in a distributed protocol.

4.3.4 Setting the Tax Schedule

How does the publisher set the tax schedule? In theory, the publisher can set the optimal tax schedule once it knows the distribution of user types. In many distributed system settings, user types are private information and users may not truthfully reveal their types to the system. A possible response is to design *strategyproof* mechanisms to induce truthful revelation by the users [25, 51]. A publisher can design an incentive compatible tax schedule such that user types can be inferred from user action.

In the case of P2P broadcasting, the user types are their bandwidth capacities (F and R), and the user actions are the actual amounts of forwarding and received bandwidths (f and r). Bandwidth capacities are static host characteristics that can be easily determined by the software agent running on the peer host, so strategyproofness is not a major concern. More interestingly, we find that accurate knowledge of user type distribution may actually be unnecessary in practice. In Section 4.5.4 we show that a fixed linear tax schedule is surprisingly robust against changes in type distributions, and good social welfare outcomes can be obtained for a wide range of values for the marginal tax rate t.

4.4 Incorporate Taxation into Protocol

This section shows how a linear taxation scheme can be implemented in a distributed protocol for video broadcasting. We leverage ideas from protocols that builds a multiple disjoint tree structure [11, 53, 19].

4.4.1 Multiple Tree Protocol

In the original multiple tree proposal, the source splits the video stream into m stripes using multiple description codec (MDC), and multicasts each stripe along a separate tree. Each peer selects *one tree* at random, and joins the trees as an *interior node*. It joins all other tree as *leaf nodes*. For example, Figure 4.4 illustrates an example structure that a multiple tree protocol constructs. Note that a peer contribute bandwidth to one of the m trees, and receive bandwidth from all the trees it joins, including the one it contributes bandwidth to.

We use MDC for a different purpose than originally proposed. MDC helps to support receiver heterogeneity, improve resilience to machine failure and network congestion. In taxation, MDC

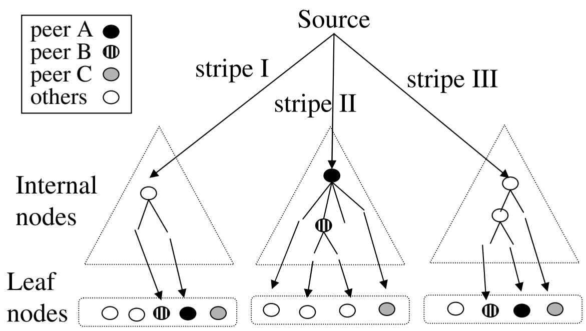

Figure 4.4: An example of a multiple disjoint tree structure.

offers peers the flexibility to contribute and receive small increments of bandwidth. Such flexibility helps us express the taxation schedule in finer granularity.

Now we attempt to incorporate the taxation semantic into this protocol and motivate the need to extend the protocol. To forward f unit bandwidth, a peer would configure the fanout of the interior node to be f. To receive r unit bandwidth, the peer would join r trees. The issue is that r depends on the available demogrant (G), and it requires global information to infer G. In the following, we describe the technique to infer G in a distributed fashion.

4.4.2 Distributed Bandwidth Allocation

Conceptually, the distributed protocol incrementally allocates r to the peers (by increasing G) until the budget is balanced. First, the heuristic assumes G is 0, and allocates the receive bandwidth to only the peers who pay tax. For peer i, this amount is equal to f_i/t. We call this the *entitled bandwidth* (r^+). A peer is entitled to receive r^+ even in a worse-case environment ($G = 0$). After all peers receive their share of r^+, the leftover bandwidth in the system is the demogrant pool. Next, the protocol iteratively increase G by 1, until the demogrant pool is exhausted. With every unit increment of G, all peers are allowed to join one more tree.

To facilitate this join order, the peer assigns a *priority* value for each of the m trees it joins. The peer marks the first r^+ joins with the highest priority. Then the peer iteratively marks all other joins with decreasing priority. Table 4.5 shows an example of priority assignment for the three peers. To receive the entitled bandwidth of 2, A sets a high priority (priority = 0) to join the first 2 trees, in which one is an interior node and the other is an leaf node. A has a lower priority (priority = 1) to join the third tree (until the demogrant becomes 1). In the case where two peers have the same priority and contend for one "spot" in a tree, an arbitration rule is needed. In our implementation, we favor the peer who has lower r, as this peer has a higher marginal utility gain in using the bandwidth.

Now that the nodes in each tree has proper priority values that reflect the tax schedule, the protocol needs to perform admission control based on the priority. Nodes that are accepted in the tree should have higher priority than those rejected. To achieve this, each interior node individually runs a *preemption* rule on the joining peers. If the fanout bound is reached, a peer with a higher priority can preempt existing peers with lower priority. Though we do not formally prove here, by induction a peer i is rejected from a given tree only if all other peers in the tree have equal or higher priority than i in the steady state. With dynamic peer environment, a peer that was previously

	f	r+	1st stripe	2nd stripe	3rd stripe
Peer A	4	2	interior node (0)	entitled leaf (0)	demogrant leaf (1)
Peer B	2	1	interior node (0)	demogrant leaf (1)	demogrant leaf (2)
Peer C	0	0	demogrant leaf (1)	demogrant leaf (2)	demogrant leaf (3)

Figure 4.5: An example depicting three peers with different forward bandwidth. Peers join each tree either as an interior node or a leaf. The numbers in bracket are their priority in receiving nth stripe. The shaded blocks are their entitled bandwidth.

preempted may become eligible. Thus a preempted peer periodically (every 30 seconds) attempt to rejoin the tree and get its fair share of the demogrant.

The peer may change its strategy (by changing f) depending in part by the available demogrant. Each peer can passively estimate G by counting the number of trees it joins. However, this estimate is not reliable due to the transient condition in the distributed protocol. To increase the accuracy of the estimate, each peer periodically (every 30 seconds) queries a subset of other peers (20 peers) about their estimates of G, and merge with its own.

4.5 Evaluation

Our evaluation seeks to answer the following three questions. For each question, we outline the evaluation methodology to answer the question.

- Does taxation yield good social welfare outcome under realistic Internet environment? We compare our proposed linear taxation scheme with two benchmark schemes. The lower bound benchmark is a Bit-for-Bit scheme, and the upper bound benchmark is a socially optimal scheme where peers are obedient (or altruistic).
- Is the proposed tax schedule effective in maximizing social welfare? We compare our fixed linear tax schedule with dynamic and non-linear tax schedules.
- What is the performance implication when incorporating taxation in a distributed overlay multicast protocol? We quantify the performance implications using three metrics: utilization, compliance, and stability.

We seek answers to these questions using two different simulation setups. The first three questions are about the fundamental efficacy of taxation, and we conduct *static simulation* with a fixed group of peers using a centralized algorithm. The last question is about implementation feasibility, and we conduct *trace simulation* using the proposed distributed protocol. This section first presents the simulation setup and the utility functions used in the evaluation, and then presents the simulation results to answer the three questions in turn.

4.5.1 Utility Functions and Example

To quantify the potential benefit of taxation, we consider a simple set of utility functions below. We acknowledge that the utility functions are based on intuition. However, we do believe the shape of

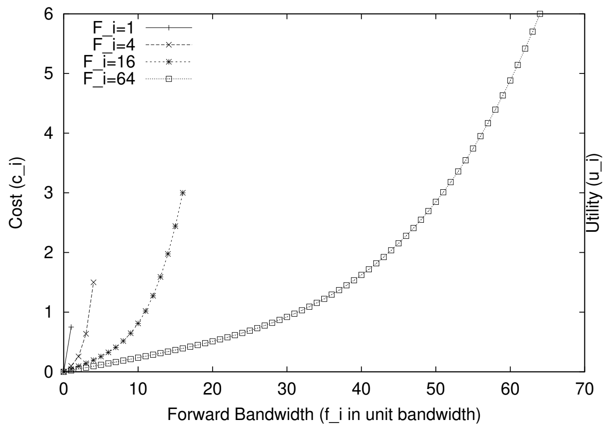

Figure 4.6: The cost as a function of forward bandwidth. The cost function depends on peer's forward capacity.

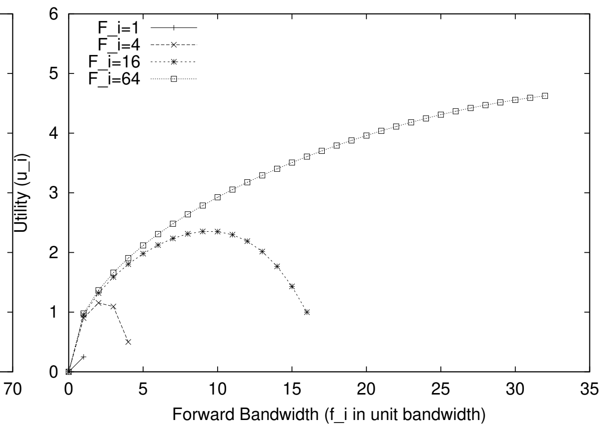

Figure 4.7: The utility as a function of forward bandwidth assuming a Bit-for-Bit scheme.

the curve (convex vs. concave) is accurate. We are in the process of collecting measurements from the real P2P environment to refine the utility functions and their parameters.

We approximate the benefit function (b_i) as a square root function to the received bandwidth (r_i). This concave function represents the diminishing benefit for the perceived video quality as the bitrate increases.

$$b_i(r_i) = \sqrt{r_i} \tag{4.4}$$

The cost function (c_i) captures the cost of forwarding data, as shown in Equation 4.5. The forwarding cost (f_i) is modeled as a fraction (p_i) of the (dollar) cost in purchasing the access bandwidth F_i. With economy of scale, the cost of F_i is concave, and is modeled as a square root of F_i. We added the α parameter to calibrate the cost function with the benefit function. α should be less than 1, indicating the desire to view the video content. In the evaluation, α is set to 0.75.

$$c_i(f_i, F_i) = \alpha * \sqrt{F_i} * p_i(f_i, F_i) \tag{4.5}$$

p_i is the fractional cost to use f_i given F_i, and the value is between 0 and 1. It is modeled as a weighed average (β) between the two components, as shown in Equation 4.6. The first component models the direct forwarding cost, where the cost of forwarding f_i is linear to the cost of F_i. The second component models the congestion cost. Congestion happens when f_i is close to F_i. Since the access links on the Internet are typically shared by different users or applications, link congestion will affect the performance (utility) of other users and applications. We model this effect as the fourth power of the linear fraction. The β parameter is set to 0.5, where both components have equal weight in the cost function.

$$p_i(f_i, F_i) = \beta * (\frac{f_i}{F_i}) + (1 - \beta) * (\frac{f_i}{F_i})^4 \tag{4.6}$$

tax schedule

f	r
0	1
2	2
4	3

outcome

	F	R	f	r	U
Peer A	10	3+	4	3	1.45
Peer B	4	3+	2	2	1.16
Peer C	1	3+	0	1	1.0

(a) Linear taxation with t=2.0, G=1: **S=1.20**

tax schedule

f	r
0	0
1	1
2	2
3	3

outcome

	F	R	f	r	U
Peer A	10	3+	3	3	1.54
Peer B	4	3+	2	2	1.16
Peer C	1	3+	1	1	0.25

(b) Bit-for-Bit with t=1.0, G=0: **S=0.98**

Figure 4.8: An example illustrating the two tax schedules and their impact on the strategy and utility of the three peers. In this example where peers are heterogeneous, taxation (with $t = 2.0$) has higher social welfare than Bit-for-Bit.

We now show with examples how peers behave with the utility functions defined above. Figure 4.6 shows the cost (c_i) as a function of f_i. Each curve represents one access capacity F_i. We make two important observations. (i) It is socially more beneficial for high capacity peers to contribute, because the per-unit cost of contributing bandwidth is lower with high capacity peers. This is shown in the figure where the curve is lower with larger F_i. (ii) Bandwidth is cheaper when a link is less congested. This is shown in the figure where the curves have convex shapes. The net effect on peer's utility is shown in Figure 4.7. It plots u_i as a function of f_i, assuming a Bit-for-Bit semantic ($f_i = r_i$). Again each curve represents one F_i. For a high capacity peer (e.g. F_i=64), the best strategy is to contribute as much bandwidth in order to receive high quality video. On the other hand, the best strategy for a low capacity peers (e.g. F_i=16) is to contribute enough bandwidth ($f_i = 9$) before the cost of congestion becomes a dominating factor.

We now provide a concrete example in Figure 4.8 to illustrate that taxation has the potential to improve social welfare over Bit-for-Bit. In this example, there are 3 peers with heterogeneous bandwidth capacities. The maximum source rate is 3 bandwidth units. Figure 4.8(a) shows a tax schedule with a marginal tax rate of 2.0 and a demogrant of 1 bandwidth unit. The outcome of the three peers are shown in the adjacent table. The outcome includes the bandwidth capacity (F, R), the chosen strategy (f, r), and the marginal utility (U). For example, A is a resource-rich peer, which can contribute up to 10 units of bandwidth. It is incentivized to contribute 4 units, in order to receive the full source rate of 3 units. This translates to a utility of 1.45. Figure 4.8(b) shows the Bit-for-Bit tax schedule and the corresponding outcome of the three peers.

Although both tax schedules collect the same tax revenue ($\sum f_i = 6$), taxation provides better social welfare than Bit-for-Bit. This improvement can be explained from two interrelated angles: (i) The cost of raising the same tax revenue is reduced with taxation. The reduction comes from a shift of tax liability from the poor (C) to the rich (A). A has lower marginal cost (1.54-1.45=0.09) of contributing one additional bandwidth unit compared to C (0.75). (ii) There is a benefit to redistributing the tax expenditure. Node C receives a higher marginal benefit from receiving the additional bandwidth unit than node A. The use of demogrant facilitates this redistribution.

4.5.2 Evaluation Environment

To realistically model the simulation environment, we use data and traces collected from several live events using a P2P broadcast system [16]. Due to space constraints, we show the results of one trace (Slashdot) in detail. The Slashdot event is the largest among all the traces, and attracted 1316 peers.

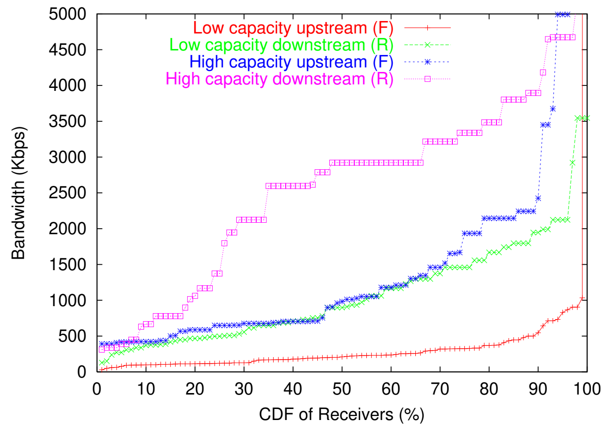

Figure 4.9: Measured TCP throughput of peers in Slashdot. Peers are categorized into low and high capacity, and the distribution is used to model environment heterogeneity in the simulation.

The mean and medium stay time is 18 minutes and 3 minutes, respectively. The trace lasts for 8 hours. The peak group size is 160. The trace contains the TCP throughput measurements (upstream and downstream) between each peer and a well-provisioned server (in our university). We use the bandwidth measurements to model the forward and receive capacity of the peers (F_i and R_i).

Static Simulation: We conduct the first two parts of the evaluation with a fixed group size (100 peers). To simulate a wide range of peer environments, we systematically vary the bandwidth capacity of the 100 peers. We consider two categories of peers, high capacity and low capacity, and model a range of peer environment by varying the *composition* of these peers. Thus, a homogeneous peer environment contains either 100% high capacity peers or 100% low capacity peers. To assign the bandwidth capacity of a peer in a category, we draw from a distribution derived from the Slashdot trace.

To derive this distribution, we first categorize each peer in the Slashdot trace using its DNS name and other access bandwidth measurements. A peer is categorized as low capacity if it is behind DSL, cable modem, or the access measurement is below T1. A peer is categorized as high capacity otherwise. Then, the bandwidth distribution is derived from all the peers under the same category. We note that in the Slashdot trace, about 20% of the peers are high capacity and 80% are low capacity. Figure 4.9 shows a CDF of peer TCP upstream (F_i) and downstream (R_i) bandwidth. We observe significant heterogeneity among peers, and the upstream bandwidth is significantly lower than the downstream bandwidth.

Trace Simulation: We conduct the last part of the evaluation by playing back the join and leave time of peers in the Slashdot trace. The peers are assigned the bandwidth capacity as recorded in the trace. The simulator captures the overlay tree changes due to peers joining and leaving the group, but not due to network congestion. So peers would switch parents only if they are preempted by other peers, or if their parents leave the group. To find a parent, a peer probes a small number of other peers (up to 5) that are the interior nodes in the tree. This limit bounds the overhead in maintaining each tree.

Parameters Common to Both Simulations: Our simulator assumes uniform delay between any

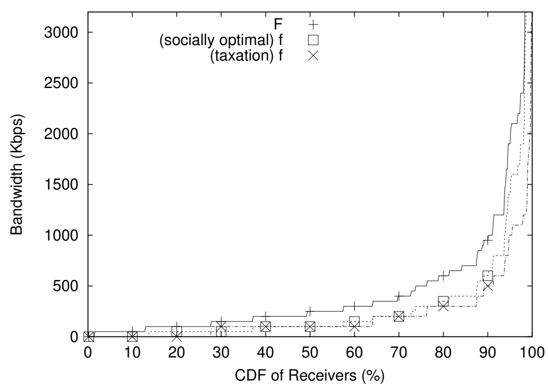

Figure 4.10: Cumulative distribution of forward bandwidth between the taxation scyeme and the socially optimal scheme. The top curve (F) indicates the forward bandwidth capacity of the peers.

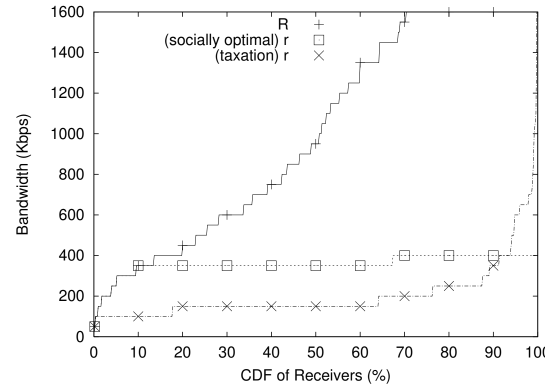

Figure 4.11: Cumulative distribution of receive bandwidth between the taxation scheme and the socially optimal scheme. The top curve (R) indicates the receive bandwidth capacity of the peers.

pair of peers and no packet loss. The maximum streaming rate is 1600Kbps. The stream is evenly divided among 32 stripes using MDC (so there are 32 trees). So each stripe (bandwidth unit) is 50Kbps.

4.5.3 Social Welfare of Taxation

We expect that taxation should improve social welfare compared to Bit-for-Bit, but it will not be *socially optimal*. We define a socially optimal scheme where peers are *obedient* in contributing bandwidth. The system designer can then dictate how much bandwidth each peer should contribute and receive to maximize overall social welfare.

For a realistic comparison, we limit the degree of obedience in the socially optimal scheme. Specifically, an obedient peer contributes only up to 6 times the full source rate. This limit avoids skewing the results if a very high capacity peer is in the broadcast system. For example, with a source rate of 1.6Mbps, a peer behind a 1Gbps link would contribute only up to 9.6Mbps. Without the limit, this one peer could virtually supply bandwidth to all other peers. This is socially desirable (sacrifice one for the benefits of all others) but not realistic. To derive a socially optimal outcome in a given peer environment, we use the following algorithm :

Socially Optimal Scheme: The algorithm iterates on the total amount of forward bandwidth (called W) in the system. Initially W is 0, and is incremented by 1 (unit bandwidth) until peers contribute all of their forward capacity. In each iteration, the algorithm minimizes the aggregate cost of raising W from the peers (called C_W) and maximize the aggregate benefit of using W among the peers (called B_W). The social welfare of W (S_W) is B_W - C_W. The socially optimal outcome then is the bandwidth distribution of W such that S_W is the highest. To get B_W, the algorithm allocates W evenly among peers (because the benefit curve is concave). To get C_W, the algorithm incrementally raises bandwidth from the peer who has the lowest marginal cost.

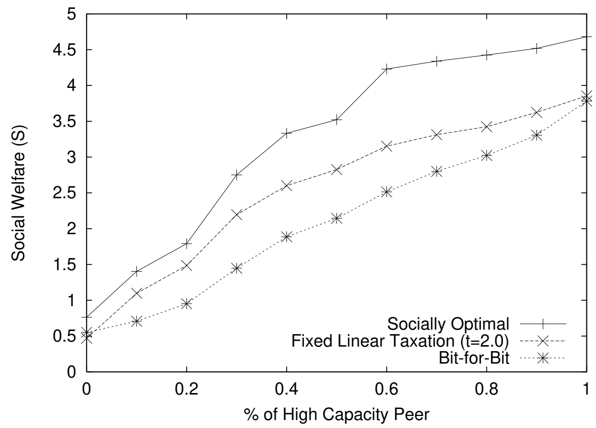

Figure 4.12: Social welfare as a function environment heterogeneity for the taxation scheme and the two benchmark schemes. Taxation has a social welfare outcome in between the two benchmarks.

We first illustrate the difference between a socially optimal scheme and a linear taxation scheme with an example. In this example, the peer environment composes of 20% of high capacity peers and 80% low capacity peers. The linear tax rate is set to be 2.0. This is an optimized choice which will be clear later in Section 4.5.4. Figure 4.10 shows the cumulative distribution of forwarding bandwidth capacity (F) and bandwidth contribution (f). The top curve is the bandwidth capacity, and the lower two curves are the bandwidth contribution for the two schemes. The higher the curve, the greater the contribution is. We make two observations. (i) For the 80% of the peers with lower forwarding capacity, peers contribute about the same in both schemes. (ii) For the 20% of the peers with higher forwarding capacity, peers in the socially optimal scheme contribute *significantly more*. For the top 5% of the peers who have significant amount of bandwidth resource, they virtually contribute almost everything to the group. This is expected because the per-unit cost is lower for the high capacity peers. As a result, it is socially more beneficial for these resource-rich peers to contribute more. In taxation, the resource-rich peers do not have incentive to contribute beyond their specified tax amount.

Another key difference is how the bandwidth resource is distributed among peers. In taxation, peers who contribute more receives more. This is not the case in the socially optimal scheme. Figure 4.11, where shows the cumulative distribution of receiving bandwidth capacity (R) and the actual bandwidth received (r) for the two schemes. The top curve is R, where 30% of the peers have the capacity to receive full source rate. The key observation is that the curve for the socially optimal scheme is almost flat line but the curve for taxation varies from 100Kbps (the demogrant) to the full source rate of 1600kbps. The socially optimal scheme results in a flat line because an even consumption of bandwidth resource is the most socially optimal outcome. In another word, bandwidth resource yields higher marginal utility when used on peers with lower receiving bandwidth. This is not possible to achieve in a taxation scheme where peers are assumed to be strategic.

We now present results with varying peer environments to illustrate the impact of heterogeneity on social welfare. Figure 4.12 shows the social welfare of the three schemes (including Bit-for-Bit) under various peer environments. 20% on the x-axis means the peer environment is composed of

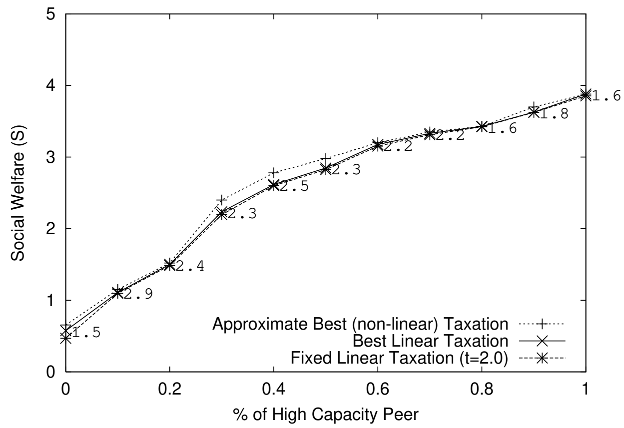

Figure 4.13: Social welfare as a function of environment heterogeneity for three different tax schedules. The proposed fixed linear tax schedule is surprisingly robust against environment changes.

20% of high capacity peers and 80% low capacity peers. Each curve represents one scheme. We make two observations. (i) Taxation is a *strongly dominating* strategy over Bit-for-Bit when peers are *heterogeneous*. For example, when the composition ratio is 20% (Slashdot environment), social welfare improves from 1 to 1.5, a 50% improvement. (ii) Taxation is a *weakly dominating* strategy when peers are *homogeneous*. This is because with similar forwarding capacity, peers will elect similar strategy in selecting f and r, and the degree of redistribution becomes minimal. (iii) The taxation scheme is still considerably worse than the socially optimal scheme. This is explained in the earlier paragraphs.

4.5.4 Effectiveness of Linear Taxation

The linear taxation schedule we propose is (i) linear and (ii) fixed (i.e. t is constant across time and event). However, this choice of tax schedule may not be effective in maximizing social welfare. In this part of the evaluation, we quantify the potential penalty of this design choice. We compare the proposed tax schedule with the following two schemes:

Best Linear Tax Scheme: In this scheme, the tax schedule is linear but the rate can be dynamically adjusted to optimize S. To find the best tax rate (t_b), the scheme varies the tax rate from 1.0 to 4.0 with an increment of 0.1 (41 possible rates), and selects the rate that maximizes S.

Approximate Best (Non-Linear) Taxation Scheme Here the tax schedule can be non-linear *and* the rate can be adjusted dynamically. This scheme approximates an ideal tax schedule that maximizes S. We are unable to come up with an algorithm to find this ideal tax schedule directly. A brute force solution seems infeasible. With a source rate of 32 unit and 41 possible tax rates, the number of possible tax schedule is as large as 32^{41}. To approximate the ideal tax schedule, the heuristic performs 20 rounds of hill climbing. At the beginning of each round, the heuristic chooses a random tax schedule. Then it iteratively adjust the 32 entries in the tax schedule, until no single adjustment can yield a higher S. Finally, the heuristic picks the tax schedule that yields the highest S among the rounds.

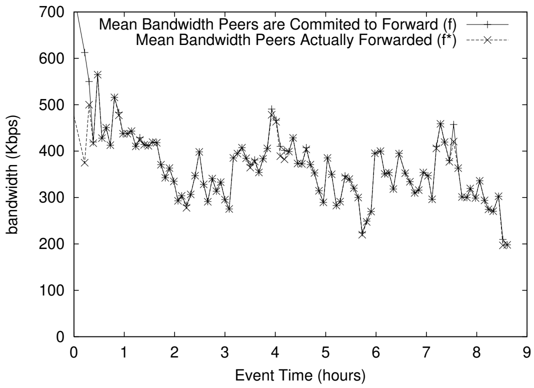

Figure 4.14: Mean bandwidth of peers vs. time in the Slashdot trace simulation.

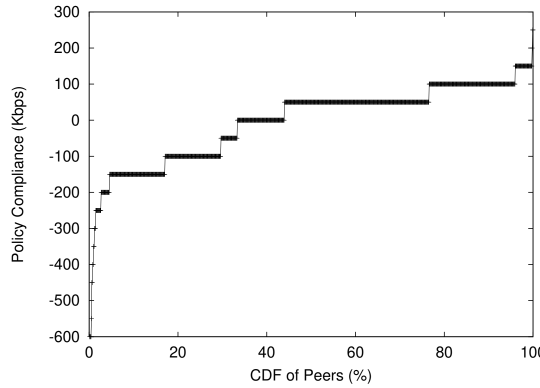

Figure 4.15: CDF of policy compliance for the peers in the experiment.

Figure 4.13 shows the social welfare of the three schemes under various peer environments. Each curve corresponds to one scheme. The lowest curve is the fixed linear taxation with a rate of 2.0. The middle curve is the best linear tax schedule, with the rate numbered for each environment. As expected, a tax schedule is more effective when it is non-linear and dynamic. However, the difference is not significant, as the three curves are very close to each other. This is an indication that a *fixed* and *linear* taxation scheme is effective under a variety of peer environments, and there is marginal benefit when tuning this parameter specific to a peer environment. Our further analysis indicates that the 2.0 linear tax rate is not a magic number. In fact, a tax rate between 1.5 and 2.5 yields similar results.

4.5.5 Distributed Protocol Performance

The distributed taxation protocol (with a tax rate of 2.0) is evaluated with the following three performance metrics:

Utilization: Ideally, an efficient protocol should utilize all the bandwidth peers contribute to the overlay. A distributed protocol cannot be as efficient because it takes some time for peers to (i) find unsaturated trees and parents, or (ii) find parents who preempt other children because the joining peers have higher priority. Figure 4.14 shows the mean bandwidth metrics of peers as a function of experiment time. The top curve is the mean bandwidth that peers are committed to forward (f). The bottom curve is the mean bandwidth that peers actually forward (f^*). The closer the two curves are, the higher the efficiency is. Our protocol is quite efficient, which utilizes at least 95% of the contributed bandwidth most of the time. There is a visible dip in utilization at the beginning of the experiment. This is because the group size is small at the same compare to the number of trees in the overlay. As a result, the aggregate forward capacity can vary greatly among trees. Thus one tree can be saturated while others are not. We currently investigate ways to dynamically balance forwarding capacity among trees.

Compliance: A good protocol should not only utilize the contributed bandwidth efficiently, but also allocate the right amount of bandwidth to peers in compliance with the taxation policy. To

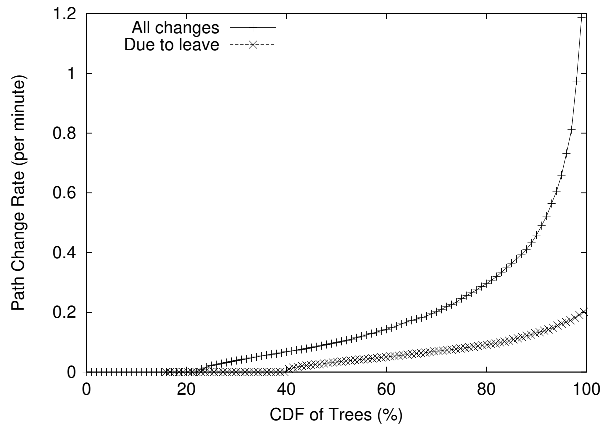

Figure 4.16: CDF of path change rate for all the trees peers participate.

measure compliance, at every time interval (5 seconds) we compare the difference between the the bandwidth a peer should receive ideally to the bandwidth the peer actually receives, i.e. $r_i - r_i^*$ (see Section 4.2 for definitions). Figure 4.15 shows a cumulative distribution of that difference for all peers during the experiment. A negative difference means the protocol allocates less bandwidth than it is supposed to. We observe that a majority of time (80%), peers see a difference no more than 100Kbps, indicating the protocol is fair in allocating bandwidth to peers. The curves show a tail as low as several hundred Kbps, indicating that peers at some point receive significant less than they should. This is mainly the artifact of the inefficiency in allocating bandwidth.

Stability: Since peers join, leave, and change their strategy dynamically (by changing f), the protocol must also dynamically adjust the bandwidth allocation (r) to the peers. Our protocol achieves high compliance and high utilization through preemption. However, preemption incurs a cost in performance. Peers that are preempted may experience transient data loss before finding another parent. Worse yet, if interior nodes are preempted, their descendants are also affected. To capture this cost, we count the rate of *path changes* for each tree that peers participate during the experiment. Some of the path changes are fundamentally unavoidable. Specifically, the path to a peer will change if any one of their ancestors leaves the group. Figure 4.16 shows the cumulative distribution of path change rate for each tree that peers participate in the experiment. We note that the total cost of implementing the policy is about twice the fundamental cost in maintaining the structure. This is shown as the difference between the two curves. The lowest curve indicates the path change rates due to peers leaving the group. A majority of the path changes due to preemption is in peers "chasing" the unused spare capacity, which can change with peers joining and leaving the group.

4.6 Related Works

Incentive Mechanisms: To our knowledge, this is among the first work to incorporate the concept of taxation into an incentive mechanism for P2P systems. Most of the prior work adopts either a token-based or a reputation-based incentive mechanism. In token-based schemes, users earn system-specific currencies such as *mojo* [82] or *karma* [78] which can be used for redeeming service. The use of market currencies, supported via micropayments, has been studied analytically in [30, 21]. In reputation-based schemes, the transactional history of each peer is used to compute its reputation [9, 40], which in turn dictates the level of service obtainable by the peer. BitTorrent is the first system that explicitly adopts the *Bit-for-Bit* scheme for P2P file sharing [20].

P2P Streaming Protocols: Most of the P2P streaming protocols today assume a "cooperative" (i.e. obedient) peer behavior. This implies that the system designer has complete control over the behavior of individual peers. Assuming peers are not strategic, these work focus on protocol designs to achieves the best outcome based on the level of cooperation considered. Bullet [39], peers are completely cooperative in contributing resource to the P2P system. The objective of both protocols is to deliver the highest multicast throughput to all peers by harnessing the bandwidth effectively. By assuming peers are obedient, the game designer (the protocols) can achieve optimal outcome that meets its objective. SplitStream [11] and CoopNet [53] assume minimal cooperation from peers and implements a *Bit-for-Bit* outcome. We leverage many of the ideas there for the protocol design. Lately, [19] devises a new protocol which allows the publisher to specify the spectrum of cooperative policy to meet different performance objectives. Finally, a reputation-based incentive mechanism has been proposed that leverages service-differentiated peer selection in many-to-one P2P streaming sessions to encourage user contribution [31].

4.7 Discussion

This is an early work that shows the promise of taxation in a heterogeneous P2P environment. However, this work is still incomplete in many ways. In the following, we highlight two important issues that are overlooked in this work. We believe further studies are needed to better understand the implication of a taxation scheme.

Utility Function: One limitation of this work is that all evaluation are conducted based on one set of utility functions (Section 4.5.1). Moreover, the parameters used in the utility functions are derived based on intuitions rather than from actual user data. We believe more studies are needed to understand the sensitivity of the taxation results on various utility functions and various user models.

We anticipate that it is not a trivial task to realistically model user behavior in P2P broadcasting environments. Our reasoning is as follows. In the taxation framework, the base currency is bandwidth, i.e. peers contribute bandwidth in exchange for bandwidth. However, users make decisions (i.e. the utility functions) based on user-perceived cost (i.e. dollar cost of bandwidth forwarding) and benefit (i.e. video quality). The difficulty is to map bandwidth to user-perceived cost and benefit. As an example, how much utility does a peer gains by receiving a 400Kbps video instead of 200Kbps? In our current utility function, the benefit function is concave to capture the diminishing returns of increased video bitrate on the perceived video quality. However, this may depend on the codecs and the type of video. For low-motion video streams (such as talking heads), even the low bitrate may yield acceptable video quality, and the additional bandwidth may not gain much. However, for high-motion video streams (such as sports events), the difference in bitrate may mean one video stream is watchable and the other is not.

Demogrant Convergence: Another issue we overlooked is the convergence properties of demogrant and its performance implication. In the current formulation of the taxation system, the demogrant (G) is determined dynamically. The advantage is that is resource is highly utilized (as shown in

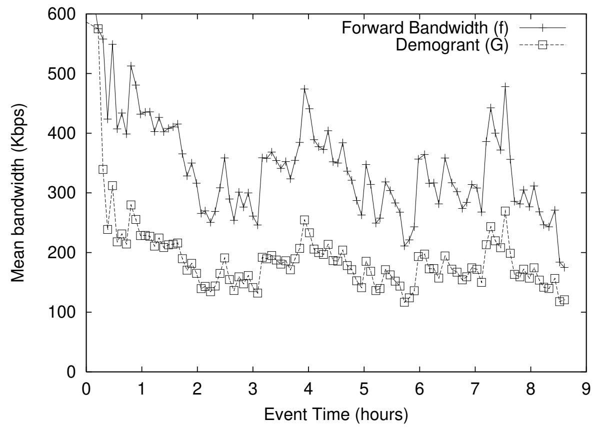

Figure 4.17: Forwarding bandwidth of peers and the resulting demogrant vs. time in the Slashdot trace simulation.

Figure 4.14). However, the key concern is that demogrant may not converge, leading to transient performance degradation. This is because when demogrant changes, the receiving bandwidth to some peer would drop. This is implemented by changing the delivery structure in the overlay that may introduce data loss during the transient.

In real system deployment, dynamic changes in peer composition drives changes to demogrant. For example, when many poor peers join the system in a close time interval, the resource becomes more scarce, causing demogrant to drop. This inter-dependence is shown in Figure 4.17, where the demogrant curve (G) follows closely with the mean forward bandwidth curve (f) in the Slashdot trace simulation. In particular, demogrant oscillates between 100Kbps and 300Kbps. Such a change in demogrant seems unavoidable because group membership is dynamic. In this work, we have captured the performance implication with the three metrics: utilization, compliance, and stability.

However, even with static group membership, demogrant still may not converge in a centralized setting. The main reason is that demogrant is not an independent variable. Recalled in Section 4.3.3, at the beginning of each round (round k), peers evaluates their utility against the current demogrant, and change their bandwidth contribution in round $k + 1$ that maximizes their utility. This change may inadvertently affect demogrant in round $k+1$, causing further changes to round $k+2$. And this cycle may repeat indefinitely. We argue that in the current problem formulation, it is very difficult to eliminate this oscillation. To avoid this oscillation in round $n + 1$, peers should predict the demogrant value in round $n + 1$, and evaluate their utility against this future demogrant. Predicting future demogrant will unlikely to be easy. In theory, such prediction can be made if the user types of all peers (i.e. bandwidth capacity and utility functions) are known. However, such knowledge is difficult to come by in practice. We believe a practical solution to avoid demogrant oscillation, even in a centralized setting, remains to be an open problem.

4.8 Summary

We identify three contributions in this chapter.

- We leverage the uniqueness of the P2P Broadcasting context and propose *taxation* as an incentive mechanism to achieve a desirable outcome. The enabling observation is the asymmetry of power, where the publisher has the power to design a taxation game and enforce the rules on the participating peers. We believe the concept of taxation is novel and has not been introduced in other P2P context.

- We show that taxation is an effective means to maximize social welfare when peers are strategic in a heterogeneous P2P environment. In an environment consisting of 80% low capacity peers, the social welfare improves by 50% compared to Bit-for-Bit.

- We demonstrate that linear taxation can be implemented efficiently in a distributed protocol with reasonable overhead. The protocol consistently utilizes above 95% of the resource and the allocation is compliant to the tax schedule within 2 bandwidth unit 80% of the time. However, this comes at a cost of structural instability due to an increase in path rate changes. We show the cost is twice the fundamental cost in maintaining the structure.

One possible objection to the taxation scheme is that mandatory taxation may displace voluntary contributions by altruistic peers. It has been hypothesized that government grants (financed through taxation) *crowd out* private philanthropy [80, 6]. However, empirical studies have provided conflicting evidence on this matter [38, 54]. Therefore, it may be prudent to design the system such that altruistic peers, if they exist, are not prohibited from contributing more than is required by the taxation scheme.

While we show the importance of altruism in peer-to-peer live broadcasting, it is not clear whether the same concept applies to other peer-to-peer applications such as file sharing. Peer-to-peer broadcasting seems unique in two ways. (i) The application "naturally" brings out altruism in people. Live broadcast is typically associated with a specific event people care about. Though altruism is also observed in file sharing, we believe people will likely be more altruistic toward an event (e.g. broadcast of Chinese New Year celebration) than a system (e.g. KaZaa). (ii) Live streaming has more stringent performance requirement. A file is still usable if it takes three times as long to download, but a video stream may not be watchable if the bitrate is three times less.

Chapter 5

Conclusion and Future Work

This chapter summarizes the contributions of this dissertation and points to future works.

5.1 Contributions

This dissertation takes on the challenge to make Internet audio/video broadcasting a *commodity service*. The dissertation provides the first complete solution that address this challenge without making any changes to the existing network infrastructure.

End System Multicast Architecture: Despite the conventional wisdom that multicast should be implemented at the IP layer, we propose an alternative architecture where end systems, and not routers, perform data replication. We call this new architecture End System Multicast. There are several benefits with this architecture. Since end systems replicate multicast data using only unicast services, there is no scaling and management concerns associated with putting state in the routers. Moreover, the deployment is immediate.

However, the key concern with this architecture is the performance penalty. In particular, End System Multicast introduces duplicated packets and incurs transient data loss when adapting to network congestion and group dynamics. To study the performance implication of this architecture, we have designed one of the first self-organizing protocols (called Narada). Our evaluation results from simulation and Internet testbed experiments indicate that the performance penalties are low from both the application and the network perspective.

Broadcast System Design, Implementation, and Deployment: To demonstrate that End System Multicast is the right architecture for video broadcast, we design, implement, and deploy a broadcast system based on this architecture. To our knowledge this is the first system with real application deployment and experience based on this architecture. The system has been in operation for over a year, and it is used by over 4000 users in 20 events. The post-mortem analysis of the event traces was positive, and provides valuable insight for future improvements. We believe our experience offers a good roadmap for others to design and deploy future Internet broadcast systems.

Incentive Mechanisms: For this architecture to succeed, the premise is that the end systems are willing to contribute bandwidth resource and participate in data replication. While in many cases it is reasonable to assume the broadcast viewers are altruistic in contributing resource, we consider the scenario in which all viewers are strategic agents, i.e. they contribute more resources only if they see clear benefits in doing so. We propose a new taxation scheme, where resource-rich peers subsidizes for the resource-poor viewers. Our simulation results indicate that taxation can achieve good social welfare without incurring a significant overhead to the system.

5.2 Reflection

So our broadcast system is out there. Any publisher can download and run the system. The overhead to install and run the system is reasonably low. The performance is reasonably good. Overall, the system seems to satisfy the needs of the publishers and the viewers. But after two years of development and deployment efforts, *why is our toolkit not widely adopted by people "like a wild fire"*? Even though periodically people come to us to inquire about using system, most of the time we must take the initiative to contact the publishers for deployment.

In this section, I take a more holistic view of this End System Multicast work, and reflect on issues that contribute to the limited deployment of this technology. I discuss issues with the technical constraints, the demand, the packaging, the economic incentives, and the application endpoint architecture.

- *Limited demand in synchronous broadcasting*: The trend for information access has been increasingly asynchronous, as seen by the popularity of TiVo and other video-on-demand services. With on-demand access, viewers are no longer constrained by the broadcast schedule, a great convenience. This trend is in contrast with broadcasting, which is inherently synchronous. Thus, there is a concern that that broadcasting is becoming a small niche with limited importance.

  However, I believe there will continue to be irreplaceable demands for synchronous broadcasting. Certain contents are far desirable to view in real-time. Examples are (i) news, such as ongoing process of war on Iraq, (ii) events, such as New Year countdown, commencement ceremony, fashion shows, rock concerts, court hearings, and sporting events, and (iii) learning, such as virtual classrooms. Students can learn by watching taped lectures. However, if they are allowed to interact with the lecturers (by asking questions), there may be a strong incentive to participate in real-time.

- *Increasing presence of NAT and firewall*: Fueled by IP address shortage and security breakins, it is increasingly common for networks to deploy NAT and firewall gateways. Our deployment experience indicate that an overwhelming 50%-70% of the user machines are behind NAT and firewall. The dissertation has explored one promising solution with Full-Cone NATs, which constitutes about 50% of all NATs deployed. However, there are still a large percentage of end systems cannot establish bi-directional communication. These end systems cannot contribute their bandwidth resource to other end systems with similar connectivity constraints. If NATs and firewalls become more and more restrictive in the near future, there could be a show-stopper for an application endpoint architecture instantiation.

  Compared to cable and satellite, our system is clearly far behind in audio/video quality, both in steady state (fidelity) and in transient (interruption). While the fidelity may be improved with better codec and higher bandwidth, the quality degradation due to interruption seems fundamental. Our broadcast system is built on the best-effort Internet, which fundamentally cannot provide end-to-end performance guarantee. Worse yet, an application endpoint architecture is even more prone to transient performance degradation due to ancestor machine failure and network congestion near the last hop, as confirmed by our experience.

As a result there is a serious concern for businesses to adopt this technology. The performance requirement is understandably high for paid viewers. Even for free viewers, this requirement may not be much lower. Consider fashion shows and company board meetings, a poor quality of the video broadcast reflects badly on the companies themselves. Without viable performance guarantees, the use of Internet broadcasting may be severely limited to amateur markets.

- *Risks of participation*: In comparing with IP Multicast, we argue that deployment based on End System Multicast is much easier because it does not require any changes in the IP routers. However, in our experience, the deployment is not a walk in the park. In an application endpoint instantiation, publishers and viewers must install our software, a risky proposition for some cautious Internet users. The fears they must overcome include: (i) The fear of spywares and Trogen horses. Currently Internet users have no easy way to establish trust with the software providers. (ii) The fear of security vulnerability through the installed software. (iii) The fear of legal and copyright infringements, particularly with the bad press of peer-to-peer networks. (iv) The fear of flooding the access link and affect others sharing the same link. We hope that emerging sandboxing computing platforms can help to restore the confidence of Internet users to install software and participate, such as Java VM and Microsoft .NET.
- *Cost of bandwidth is dropping*: In Section 2.6, we argue for an application endpoint architecture because of the bandwidth cost in delivering video. However, as technology advances, the cost of bandwidth will continue to drop. Thus it is possible that in the future, the cost is low enough that a third party provider can make broadcast as a free commodity service, such as email. In return, the third party provider can recoup the operating cost by means such as advertisement. When this happens, the need for an application endpoint architecture is potentially marginalized.

Despite all of these concerns, I still believe an application endpoint architecture for video broadcasting will continue to strive in the foreseeable future. The cost of bandwidth will remain high, and so is the infrastructure-based approaches. As a result, there will continue to be a strong incentive for publishers and viewers to use an application endpoint solution, even though the performance is lower and the risk is greater. In addition, there are still ample rooms for exciting research to improve performance in the best-effort heterogeneous Internet environment to address the NAT and firewall issues.

5.3 Future Directions

This section suggests a list of future research directions.

- *On-demand video distribution*: Video on-demand and live video broadcasting share one important characteristic: both require the same amount of bandwidth irrespective of the online/offline nature of the video data. As a result, many aspects of this dissertation apply to the video on-demand. From the architecture perspective, the cost benefit in using an application endpoint architecture still stands. From the protocol perspective, both applications require an efficient overlay multicast protocol optimized for bandwidth. Moreover, the issues of NAT/firewall and incentive will likely be the same, and so are the solutions.

  However, the key difference is that video on-demand does not require real-time data delivery. As a result, the detail solutions may be quite different. Since data is no longer in real-time, end systems can buffer data. Buffering greatly reduces the impact of transient performance degradation due to host dynamics and network congestion. However, buffering comes at a cost of additional protocol complexity. Since the amount of data in the buffer depends on the join time and the receiving rate, end systems will have different amount of data. Thus, the protocol must be intelligent in finding parents who have the desired data in their buffers.

- *Compare distributed and centralized protocols*: There seems to be a religion among network researchers that centralized protocols should be avoided at all cost. Since the beginning of this work, we subscribed to the same religion without giving a second thought. Narada protocol is completely distributed, as well as most other protocols in the research literature. I believe the cost/benefit tradeoffs with a centralized approach are far from clear, and deserve to be studied in detail. Moreover, the study should consider the requirements and circumstances of the system deployment.

  A distributed protocol typically has two potential benefits: robustness (no single point of failure) and load balancing (no server load bottleneck). In the context of video broadcasting, these benefits may not be insignificant. There is no gain in robustness since the source is inherently a single point of failure. Also, it is questionable whether the server would become a bottleneck. CoopNet [53] shows that a commodity server can serve at least tens of thousands of machines. On the other hand, a distributed protocol has many technical challenges and performance drawbacks. Even in a relatively high dynamic environment such as video broadcasting, it is unclear whether any protocol can scale up to a large group size while retaining its efficiency.

- *Video broadcast and social science*: In this dissertation, we use the broadcast system as a research vehicle for networking. However, we believe it has value extending into other areas of research. For example, one potential direction is in social science: Is there a demand for *interactivity* in video broadcast and what are effective means of interactivity? Unlike traditional TV broadcast, Internet broadcast provides opportunities for viewers to actively engage in the event. We believe interaction among viewers are new and valuable social capital that did not previously exist in traditional TV broadcast. If interactivity is used effectively, it can enhance the viewing experience, create positive feedback, and grow into a virtual community.

- *Selfish routing vs. coordinated adaptation*: Imagine the scenario where the application endpoint architectures are widely adopted, and many groups of people run the video broadcast systems simultaneously. Will these groups co-exist happily without any explicit coordination among themselves? Consider a simple example where a DSL user participates in two video broadcasts. How should the two broadcast overlays share the bottleneck resource of the DSL bandwidth?

The key observation is that links in the overlay do not represent *independent* pieces of network resource, as two different overlay links may share the same physical links in the underlying network. So far we have abstracted out this inter-dependency and mark the resource (e.g. available bandwidth) of each overlay link independently. This abstraction so far works well for an overlay network supporting just one multicast group, but the abstraction may break when supporting multiple groups, where traffic generated by these groups is more likely to contend for the same physical resources. In this case, an explicit coordination among groups may be necessary to optimize the use of limited bandwidth resources.

- *Scalability to group size*: One important question that we have not been able to pursue fully in our deployment efforts is scalability. There are external factors that make it difficult to control the group size or user participation. Availability of compelling content plays a key role. The broadcasts that we have conducted so far have been for technical content which tends to attract a limited audience. Content that may attract a wider audience, for example, movies and entertainment, is not freely available and often has copyright limitations. It remains an open question whether an application endpoint architecture can scale to very large group size in actual deployment.

Bibliography

[1] E. Adar and B. A. Huberman. Free-riding on Gnutella, 2001. First Monday 5(10).

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

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

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

[5] S. Banerjee, S. Lee, B. Bhattacharjee, and A. Srinivasan. Resilient multicast using overlays. In *Proceedings of ACM Sigmetrics*, June 2003.

[6] T. Bergstrom, L. Blume, and H. Varian. On the private provision of public goods. journal of public economics 29: 25-49, 1986.

[7] R. Blahut. Theory and practice of error control codec. addison wesley, ma, 1994.

[8] E. Bommaiah, A. McAuley, R. Talpade, and M. Liu. Amroute: Adhoc multicast routing protocol. Internet draft, Internet Engineering Task Force, August 1998.

[9] C. Buragohain, D. Agrawal, and S. Suri. A game theoretic framework for incentives in p2p systems. In *International Conference on Peer-to-Peer Computing*, 2003.

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

[11] 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.

[12] M. Castro, P. Druschel, A. Kermarrec, 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.

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

[14] Y. Chawathe. Scattercast: An architecture for Internet broadcast distribution as an infrastructure service. Fall 2000. Ph.D. thesis, U.C. Berkeley.

[15] Y. Chu, J. Chuang, and H. Zhang. A case for taxation in peer-to-peer streaming broadcast. In *ACM SIGCOMM Workshop on Practice and Theory of Incentives and Game Theory in Networked Systems (PINS)*, August 2004.

[16] Y. Chu, A. Ganjam, T. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early Experience with an Internet Broadcast System Based on Overlay Multicast. In *USENIX Symposium on Internet Technologies and Systems*, June 2004.

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

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

[19] Y. Chu and H. Zhang. Considering altruism in peer-to-peer internet streaming broadcast. In *ACM NOSSDAV*, June 2004.

[20] Bram Cohen. Incentives build robustness in bittorrent. In *1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[21] C. Courcoubetis and P. Antoniadis. Market models for p2p content distribution. In *1st International Workshop on Agents and Peer-to-Peer Computing*, July 2002.

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

[23] S. Deering and R. Hinden. Internet protocol, Version 6 (IPv6) Specification, rfc-2460, December 1998.

[24] C. Faloutsos, M. Faloutsos, and P. Faloutsos. On power-law relationships of the internet topology. In *Proceedings of ACM Sigcomm*, August 1999.

[25] J. Feigenbaum and S. Shenker. Distributed algorithmic mechanism design: Recent results and future directions. In *6th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications, pp. 1-13*, 2002.

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

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

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

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

[30] P. Golle, K. Leyton-Brown, and I. Mironov. Incentives for sharing in peer-to-peer networks. In *ACM Conference on Electronic Commerce*, 2001.

[31] A. Habib and J. Chuang. Incentive mechanism for peer-to-peer media streaming. In *IEEE International Workshop on Quality of Service (IWQoS'04)*, 2004.

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

[33] 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.

[34] 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.

[35] 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.

[36] 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.

[37] S. King, R. Fax, D. Haskin, W. Ling, T. Meehan, R. Fink, and C. E. Perkins. The case for IPv6, 1999. Internet Draft.

[38] B. R. Kingma. An accurate measurement of the crowd-out effect, income effect, and price effect for charitable contributions. journal of political economy 97: 1197-207, 1989.

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

[40] H.T. Kung and C.-H. Wu. Differentiated admission for peer-to-peer systems: Incentivizing peers to contribute their resource. In *1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[41] K. Lai and M Baker. Nettimer: A Tool for Measuring Bottleneck Link Bandwidth. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, March 2001.

[42] B. N. Levine, D. B. Lavo, and J. J. Garcia-Luna-Aceves. The case for concurrent reliable multicasting using shared ACK trees. In *Proceedings of ACM Multimedia'96*, November 1996.

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

[44] J. Liebeherr and B. S. Sethi. A scalable control topology for multicast communications. In *Proceedings of IEEE Infocom*, April 1998.

[45] 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.

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

[47] S. McCanne, V. Jacobson, and M. Vetterli. Receiver-driven layered multicast. In *Proceedings of ACM SIGCOMM*, August 1996.

[48] S. Michel, K. Nguyen, A. Rozenstein, L. Zhang, S. Floyd, and V. Jacobson. Adaptive web caching: towards a new global caching architecture. *Computer Networks and ISDN Systems*, November 1998.

[49] James Mirrless. An exploration in the theory of optimum income taxation, review of economic studies 38 (1971): 175-208.

[50] A. Mohr, E. Riskin, and R. Ladner. Unequal loss protection: Graceful degradation of image quality over packet erasure channels through forward error correction. In *IEEE JSAC*, June 2000.

[51] C. Ng, D. Parkes, and M. Seltzer. Strategyproof computing: Systems infrastructures for self-interested parties. In *1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[52] 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 *Proceedings of IEEE Infocom*, April 2003.

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

[54] A. A. Payne. Does the government crowd-out private donations? new evidence from a sample of non-profit firms. journal of public economics 63: 323-45, 1998.

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

[56] 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.

[57] R. Perlman, C. Lee, T. Ballardie, J. Crowcroft, Z. Wang, T. Maufer, C. Diot, J. Thoo, and M. Green. Simple multicast: A design for simple, low-overhead multicast. Internet Draft, Internet Engineering Task Force, March 1999. Work in progress.

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

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

[60] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Application-level Multicast using Content-Addressable Networks. In *Proceedings of NGC*, 2001.

[61] Real broadcast network. http://www.real.com/.

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

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

[64] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. In *ACM Computer Communication Review*, January 1997.

[65] J. Rosenberg, J. Weinberger, C. Huitema, and R. Mahy. STUN - Simple Traversal of UDP Through Network Address Translators. IETF-Draft, December 2002.

[66] 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.

[67] 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.

[68] F.B. Schneider. Byzantine generals in action: Implementing fail-stop processors. *ACM transactions on Computer Systems, 2(2)*, pages 145–154, 1984.

[69] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications. RFC-1889, January 1996.

[70] The Search for Extraterrestrial Intelligence SETI@home. http://setiathome.ssl.berkeley.edu/.

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

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

[73] Nicolas Stern. On the specification of models of optimum income taxation. journal of public economics 6 (july-august 1976): 123-162.

[74] 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.

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

[76] Understanding Universal Plug and Play. Microsoft White Paper.

[77] M. Veseth. Public finance, reston publishing, 1984.

[78] V. Vishnumurthy, S. Chandrakumar, and E. Sirer. Karma: A secure economic framework for p2p resource sharing. In *1st Workshop on Economics of Peer-to-Peer Systems*, June 2003.

[79] 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.

[80] P. G. Warr. Pareto optimal redistribution and private charity. journal of public economics 19: 131-8, 1982.

[81] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. In *OSDI02*, pages 255–270, Boston, MA, December 2002.

[82] B. Wilcox-O'Hearn. Experiences deploying a large-scale emergent network. In *1st International Workshop on Peer-to-Peer Systems*, March 2002.

[83] R. X. Xu, A. C. Myers, H. Zhang, and R. Yavatkar. Resilient multicast support for continuous-media applications. In *Proceedings of NOSSDAV'97*, May 1997.

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

[85] 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*, April 2001.