University of California
Los Angeles

Universal IP Multicast Delivery

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Beichuan Zhang

2003

The dissertation of Beichuan Zhang is approved.

Mario Gerla

Richard R. Muntz

Frederic Paik Schoenberg

Lixia Zhang, Committee Chair

University of California, Los Angeles

2003

To Mo Xiao

Table of Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my advisor, Lixia Zhang, for her wisdom, generosity, encouragement and support during my graduate studies. Her mentorship will always serve me as an example. I would also like to thank Sugih Jamin for his tremendous help and guidance during this dissertation research. I am grateful to my colleagues in UCLA Internet Research Lab for their technical help and friendship.

My parents, my brother Beilin, and my sister-in-law May provide me their unwavering support and love. There is no adequate way to express my appreciation to my wife Mo. She has been my inspiration and my comfort, my best friend and my closest family.

Vita

1972 Born, Beijing, China

1995 B.S. (Chemistry), Peking University, China

1997 M.S. (Chemistry and Biochemistry), University of California, Los Angeles

Publications

Yixin Jin, Beichuan Zhang, Vasileios Pappas, Sugih Jamin, Lixia Zhang, *"DIP: A Scalable Soft-State Protocol for Large Scale Data Dissemination in IDMaps System,"* Appeared in the Eighth IEEE Symposiums on Computers and Communications (ISCC), Kemer-Antalya, Turkey, June, 2003

Beichuan Zhang, Sugih Jamin, Lixia Zhang, *"Universal IP Multicast Delivery,"* Appeared in the Fourth International Workshop on Networked Group Communication (NGC), Boston, MA, October, 2002

Beichuan Zhang, Sugih Jamin, Lixia Zhang, *"Host Multicast: A Framework for Delivering Multicast to End Users,"* Appeared in IEEE INFOCOM, New York City, NY, June, 2002

Daniel Lihui Gu, Guangyu Pei, Henry Ly, Mario Gerla, Beichuan Zhang , Xiaoyan

Hong, *"UAV Aided Intelligent Routing for Ad-Hoc Wireless Network in Single-Area Theater,"* Appeared in IEEE Wireless Communications and Networking Conference (WCNC), Chicago, IL, September, 2000

Beichuan Zhang, Bing Zhang, Irene Wu, *"ETCP: Extended TCP for Mobile IP Support,"* Internet Draft, work in progress, 1998

Abstract of the Dissertation

Universal IP Multicast Delivery

by

Beichuan Zhang

Doctor of Philosophy in Computer Science

University of California, Los Angeles, 2003

Professor Lixia Zhang, Chair

A ubiquitous and efficient multicast mechanism is essential to the success of large-scale group communication applications. Traditional IP Multicast is implemented at network layer on routers, achieving great transmission efficiency but also posing significant challenges to deployment. End-host Multicast moves multicast functionality to transport or application layer on end hosts to have minimal deployment barrier, but incurs performance penalty due to duplicate packet transmission and inefficient routing. Between them, there are a spectrum of other multicast schemes trying to make better trade-off between system performance and deployability. Nevertheless, the lack of ubiquitous multicast support on the current Internet hinders the development of multicast applications, which in turn reduces incentives for multicast deployment.

In this dissertation research, we designed and implemented the Universal Multicast (UM) framework to provide ubiquitous multicast delivery service on the Internet. Unlike previous work, we do not implement multicast functionality at a fixed place in the network for a particular deployment stage; instead, UM is a general framework within which applications can be provided with IP Multicast delivery immediately, and infrastructure support for multicast can gradually

spread from the network edges into the core. Therefore the trade-off between system performance and deployability is not fixed by the design, but can evolve over time driven by user demand for the service.

Our approach is to build an end-host overlay to connect IP-Multicast 'islands' by adaptive unicast tunnels, and use native IP Multicast within each island. The design consists of three major components: Host Multicast Tree Protocol (HMTP) for inter-island routing, Host Group Management Protocol (HGMP) for intra-island management, and a user Agent implementing the functionalities on end hosts. UM takes full advantage of infrastructure support where available and utilizes End-host Multicast where needed. It is fully distributed, self-organized, and independent from underlying multicast routing protocols. We evaluated the system performance through simulation, implemented a prototype and tested with existing multicast applications.

CHAPTER 1

Introduction

Multicast is sending same data packets to multiple receivers, as opposed to unicast, in which there is only one receiver. Multicast is widely used in applications involving one-to-many or many-to-many communication, such as teleconferencing, distant learning, multimedia distribution, peer-to-peer file sharing, multi-player gaming and so on. On the current Internet, the dominant approach to implement multicast is using unicast to send data repeatedly from the source to every receiver (Fig. 1.1). This *repetitive unicas*t approach is popular because it does not require any functionality other than the existing unicast connectivity provided by the Internet. However, its scalability is limited. Sending the same packet multiple times consumes more bandwidth, especially on the source's last-hop link, where the bandwidth consumption is proportional to the number of receivers. The source also has to know all the receivers in order to send, which often leads to a centralized design for applications with multiple sources: packets are unicast to a central server first, then the server, who knows all the receivers, multicast the packets using repetitive unicast. This design suffers from traffic concentration and single-point failure at the central server. Therefore, most current applications, using repetitive unicast, are limited to small groups. For the success of large-scale group communication applications, a ubiquitous and efficient multicast mechanism is needed.

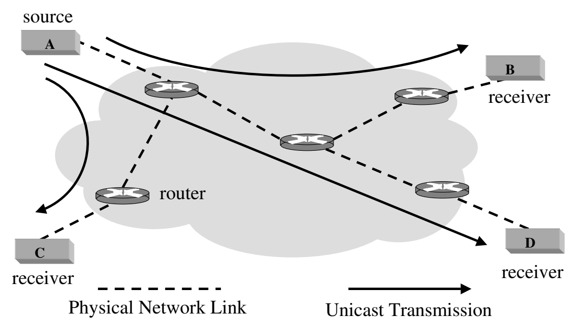


Figure 1.1: Repetitive Unicast

1.1 IP Multicast

1.1.1 Design

In late 1980's, Stephen Deering proposed IP Multicast in his seminal work [DC90]. IP Multicast uses a class D IP address (224.0.0.0 - 239.255.255.255) to identify each multicast group. Receivers join a group by reporting its interest to the edge multicast router; senders send to the group by putting the group address as the packet's destination IP address; routers replicate packets at appropriate branching points in the network and forward to all receivers (Fig. 1.2). The data delivery is very efficient because one physical link carries the same packet for only once, regardless of the number of receivers or number of senders. IP Multicast adopts an *open group model*, which uses the group address as a logical identifier of the group. Given the group address, any source can send packets to the group, and any receiver can 'tune in' to receive, without any coordination between them in advance. Tasks like tracking group membership, duplicating packets, and forwarding packets are done by routers within the network. Therefore, IP Multicast can scale to very large groups, and is very convenient for resource discovery (Chapter 4).

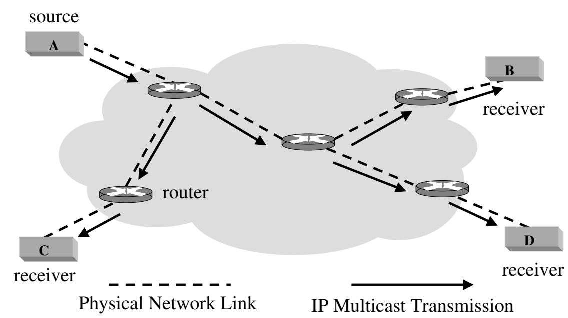


Figure 1.2: IP Multicast

The architecture of IP Multicast consists of three parts. First, the host operating system must support multicast host extension, being able to send and receive packets destined to class D addresses. Second, the edge multicast router runs Internet Group Management Protocol (IGMP) [Fen97] to poll group membership in its local area network, while hosts report their interests via IGMP too. Third, multicast routers run a multicast routing protocol to build data delivery tree(s) among them. Depending on which protocol is in use, it can be a single tree shared by all sources, or reverse shortest-path trees rooted at each source.

Several multicast routing protocols have been proposed, such as DVMRP [WPD88], MOSPF [Moy94], CBT [BFC93] and PIM-SM [EFH98]. DVMRP is a flood-and-prune protocol building reverse shortest-path trees. The source first floods the entire network with a few data packets. A router accepts a packet only if it arrives at the interface used as the outgoing interface for the shortest path to the source. Otherwise the packet is dropped, and a prune message is sent back. Routers without downstream group members also send prune messages upstream. The result is a reverse shortest-path tree, with source as the root, and receivers as the leaves. One drawback of per-source tree protocol like DVMRP is that routers have to maintain routing state for each (group, source) pair, making it not scalable for groups with many sources. PIM-SM improved the scalability based on the concept of rendezvous point (RP). RPs are pre-defined routers in the network. A list of candidate RPs is flooded to the network, and edge routers use a hash function to pick an RP for each multicast group. Receivers join a group by sending join messages to the group's RP, forming a uni-directional shared-tree rooted at the RP. Sources send data packets to RP, who then forward them to all receivers along the shared-tree. Since all sources share the same forwarding tree, routers now keep only one route entry for each group, regardless of the number of sources.

Though PIM-SM scales better than DVMRP, it is still not suitable for applications across multiple network domains, partly because network operators do not like to rely on RPs in someone else's network, and partly because the flooding of candidate RP list limits its usage in large networks. The current solution for deploying inter-domain multicast routing is using PIM-SM, MBGP [BC88] and MSDP [MF01] together. MBGP is a BGP extension which exchanges multicast routes and policies among different Autonomous Systems (AS). MSDP allows a multicast group to have multiple RPs. It connects RPs in different ASs by TCP, announces existence of sources, and disseminates group-to-RP mappings. In the future, inter-domain multicast routing is expected to move to BGMP and MASC [KRT98]. BGMP connects ASs with group members by a bi-directional shared-tree rooted at the core AS of the group. It relies on MASC to dynamically assign core ASs to multicast groups.

IP Multicast's address allocation is different from that of unicast. The fundamental problem is that the multicast address space is small compared with the number of potential groups. Since most groups are not permanent, dynamically assigning and recollecting multicast addresses can support much more groups than static allocation. Currently, running applications announce the multicast addresses they are using and the expected time to finish; new applications collect all addresses being used and randomly pick an unused multicast address for themselves. This approach does not work well in large-scale networks with lots of groups, because the probability of address collision increases fast. IETF MALLOC working group has designed MAAA (Multicast Address Allocation Architecture) [THE00] to dynamically allocate multicast addresses in a top-down manner along the network hierarchy. MAAA uses three protocols: MASC [REG00] for inter-domain address allocation, AAP [HH00] for intra-domain allocation, and MADCAP [HPS99] for allocating addresses to client hosts. MAAA has not yet been widely implemented or deployed on the Internet.

1.1.2 Deployment

IP Multicast requires routers run multicast routing protocols and forward packets. Because of the Internet's ever increasing size and the lack of central administrative control, upgrading all routers overnight is not possible. A partial deployment strategy, known as MBone (Multicast Backbone) [Eri94], is used to facilitate gradual deployment of IP Multicast. Under this strategy, individual networks deploy IP Multicast within their own domains, and then set up IP-encapsulated *unicast tunnels* to other IP-Multicast-enabled networks, thus forming the MBone. Each tunnel is a virtual link that may cross several physical links and unicast-only routers. The multicast router at one end of the tunnel encapsulates multicast packets and unicast to the other end, which will decapsulate the packets and multicast to local network. With MBone, IP-Multicast-enabled networks can communicate with each other and run multicast applications, even if there are unicast-only networks sitting in between. It was hoped that as more networks

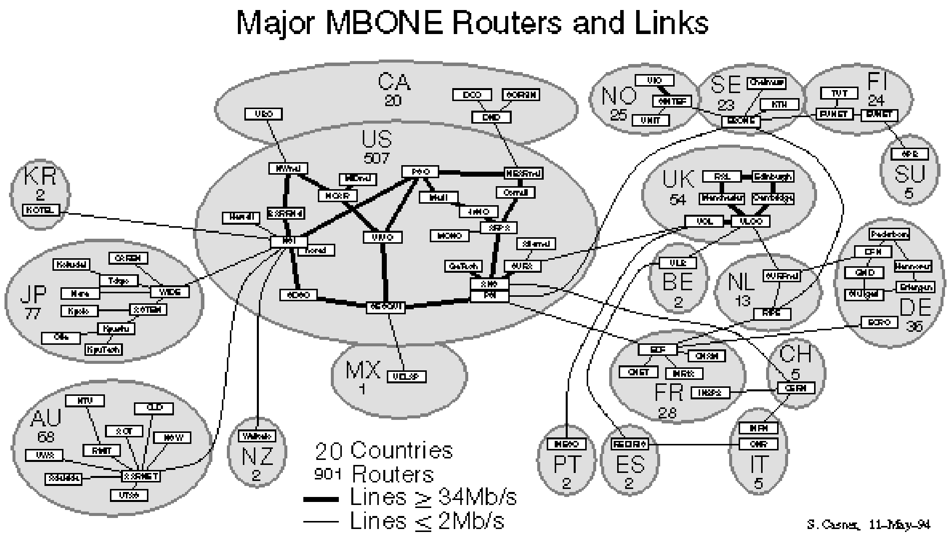


Figure 1.3: MBone topology

deploy IP Multicast and join the MBone, eventually the MBone will cover the entire Internet, and multicast delivery becomes ubiquitous.

MBone enjoyed fast growth in its early days. In March 1992, MBone carried its first wide-area event by delivering audio from IETF meeting to 20 sites worldwide [CD92]. By 1995, MBone had more than 10,000 users in 30 countries doing collaborative work, its traffic doubled every eight months, and it had carried webcast for events like space shuttle launch and Rolling Stone concert [Kah95]. MBone also became a testbed for new multicast protocols and applications. Upper level protocols (e.g., RTP [SCF96], SRM [FJL97]) were designed and implemented, Internet tele-conferencing tools were developed, and new commercial equipments and softwares were tested on the MBone. Fig. 1.3 [Cas95] shows major MBone routers and links at May 1994.

However, MBone gradually lost its momentum since late 1990's, mainly because ISPs have been reluctant to deploy IP Multicast in commercial networks.

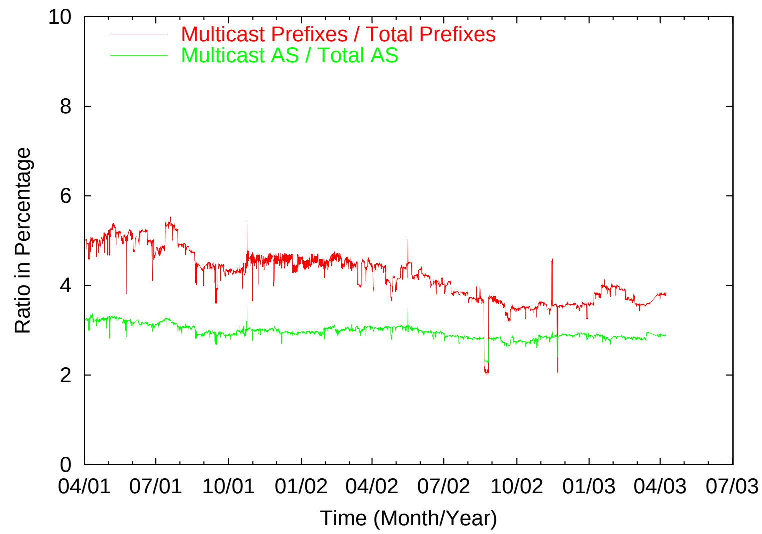


Figure 1.4: IP Multicast coverage on the Internet

Fig. 1.4 shows the IP Multicast coverage on the Internet in percentage of multicast-enabled ASs and address prefixes. The data was collected from MBGP announcements received at AS 16517 [Eub03]. Over the last two years, IP Multicast coverage stays around 4% of the Internet without significant change.

According to [DLL00], network operators have several concerns in deploying IP Multicast:

1. The infrastructure is complex. To operate at the Internet scale, IP Multicast needs routing protocols, address allocation protocols, and several other supporting protocols, which make it much more complex than the unicast infrastructure. Furthermore, upper-layer protocols that implement reliability and security are also more complex than their unicast counterparts. The complexity makes implementation, deployment and management more costly.

2. An effective and scalable address allocation mechanism is still missing. The

current random allocation does not scale well, while the MAAA has not yet begun deployment. If two groups accidentally choose the same address, the damage is more serious to commercial applications than to research applications.

3. The open group model is not adequate. It does not address group management issues, such as authorization for group creation, receiver authorization, sender authorization, and service billing, which are important for commercial operations.

As the Internet continues its fast growth, more bandwidth and faster computers become available to end users, and make more large-scale group applications possible. Nevertheless, we have not seen proliferation of multicast applications as predicted by many people in mid-1990's. Application developers wait for IP Multicast to be deployed before engaging in serious development; while network operators want to see real 'killer' applications before committing resources to deploying IP Multicast. The result is a deadlock between application development and network service deployment, even though there is increasing need for multicast applications from end users.

Fifteen years after Deering's original paper, IP Multicast coverage on the Internet is still spotty at best, limited to research community and a handful service providers. Today, IP Multicast is implemented in most host operating systems and routers, but not widely enabled, and most ISPs do not offer multicast service to end users. The lack of ubiquitous multicast connectivity limits the development of multicast applications, which in turn reduces the incentive for multicast deployment.

1.2 Alternate Multicast Schemes

To meet application's increasing needs for multicast delivery service, many alternate schemes have been proposed to bring multicast to the Internet. Based on their views of what causes the IP Multicast's deployment problem, these alternate schemes can be categorized as either router-based or host-based.

1.2.1 Router-based

The router-based schemes attributes IP Multicast's deployment problem to its design flaws. The idea is to fix the design, and hopefully the deployment will follow. Among them, Simple Multicast [PLB99] and Source-Specific Multicast (SSM) [HC00] are typical efforts of simplifying the infrastructure.

Simple Multicast builds a bi-directional shared-tree for each group. The tree is rooted at a *core* router. The core and the group address together identify a group. Therefore, the allocation of group addresses becomes a local decision done by the core router only. Core placement and discovery are taken out of the basic multicast protocol and left to applications to handle. For example, users may find out the core and the group address by email, web or DNS.

SSM simplifies the protocol design by restricting each group to have only one single source. The address allocation is solved similarly to Simple Multicast, as every group is identified by the source and the group address together. The source can handle authorization, accounting and billing. A shortest-path tree rooted at the source is enough for packet delivery. Because of the single-source restriction, SSM is best suited for Internet-wide broadcast applications, in which a single dominant source is often the case. Multi-source support is left to applications to handle, e.g., using multiple groups.

There are many other router-based schemes (e.g., REUNITE [SNZ00], HBH [CFD01], Aggregated Multicast [FCG01] etc.), trying to improve or simplify IP Multicast on a specific aspect. However, they all share two major disadvantages. First, they are still router-dependent, which means the deadlock between application development and service deployment still exists. Second, they lack IP Multicast's installation base, built by a decade's efforts from both academia and industry. Taking into account that today's Internet is much more established than fifteen years ago, a new router-based multicast scheme will have harder time to overcome the inertia and gain momentum in deployment. For example, SSM is designed to be integrated into existing PIM-SM protocol with small changes, and is backed by some major vendors and ISPs, but it still has not established significant presence on the Internet.

1.2.2 Host-based

The host-based schemes attributes the multicast deployment problem to its router dependency, not design flaws if any. The solution is to move multicast functionality from routers to hosts. Routers only need to support unicast, while hosts collaborate with each other to duplicate and forward packets. This form of multicast is also often referred as *application-layer multicast*. Depending on where the hosts are, these schemes can be further divided as End-host Multicast and Multicast Server Overlay.

In End-host Multicast, group members, as end hosts, are organized together to realize multicast delivery. For example, in Fig. 1.5, source A sends packets to C and D, the D duplicates and forwards packets to B. A basic assumption is that group members are willing to share bandwidth and computing power to facilitate the group communication, which is reasonable to most group applications;

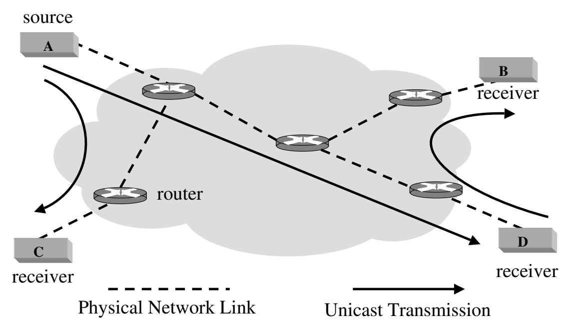


Figure 1.5: End-host Multicast

otherwise, users would not join the group at the first place. Recent advances in Internet access technology bring more bandwidth to end hosts, making them more suitable for packet forwarding. Compared with repetitive unicast, End-host Multicast is more scalable because the workload is shared by all group members, instead of only the source. Compared with router-based schemes, End-host Multicast's deployment, which is installing a software at end-hosts, is much easier and done by users. However, its routing is less efficient, and duplicate transmission on physical links is inevitable. End hosts are also less stable and less trustworthy than routers since they are owned and operated by individual users.

Multicast Server Overlay is an effort to improve End-host Multicast's performance and stability. It consists of a large number of dedicated servers placed all over the Internet. The servers are deployed by a service provider and under the same administrative control. The overlay receives packets from data sources, duplicates and forwards packets among the overlay servers, and eventually delivers to client hosts. Multicast server overlay can provide better quality of service, but at the expense of deployability. In stead of waiting for ISPs to deploy IP Multicast, users have to wait for some overlay service providers to deploy the overlay before getting multicast connectivity.

1.3 Universal Multicast Framework

Though dozens of multicast schemes have been proposed, most Internet users still have no multicast connectivity due to lack of deployment. Since deploying a network protocol or service is considered a task for network operators, deployment issues are often overlooked by protocol designers. Deployability was not much of a concern in Internet's early days, when it was a cooperative network for the research community. But today's Internet is mainly composed of commercial networks under different administrative control. As a result of the success of existing Internet and its applications, adding new services, including multicast, into the network becomes much more difficult since the deployment must be justified economically in advance. A protocol designed without explicit deployability consideration will have a hard time to be seen on the real Internet. This dissertation work proposed a new multicast infrastructure, Universal Multicast Framework, to tackle the multicast deployment problem and bring ubiquitous multicast to the Internet.

1.3.1 Deployability versus Performance

A fundamental question in network protocol design is where to implement certain functionality. According to the *End-to-End argument* [SRC84], a service should be pushed to end hosts if possible, unless the performance gain of implementing it within the network outweighs the cost associated with the deployment. However, evaluating IP Multicast's performance gain against its deployment cost is not an easy task. First, it depends on the size of the multicast group. Previous work [CS01] [CA01] have shown that bandwidth saved by IP Multicast over repetitive unicast increases as the group size grows. IP Multicast is indispensable for applications like Internet broadcasting, but repetitive unicast is probably adequate

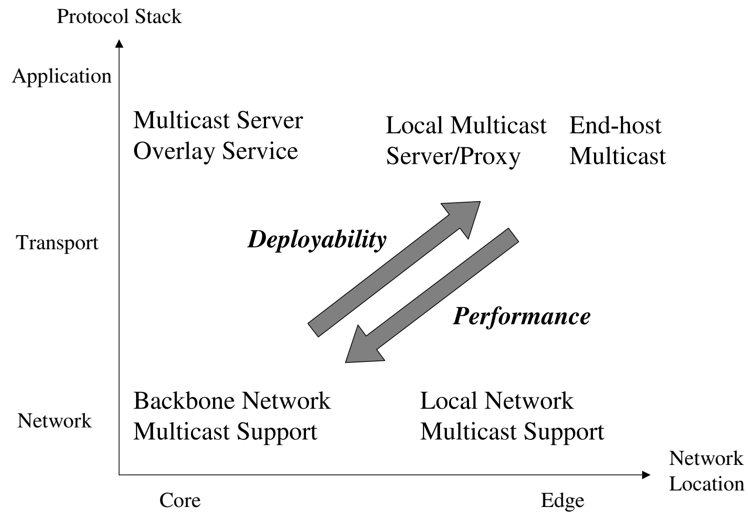


Figure 1.6: Deployability v.s. Performance

for applications like small-scale chat rooms. Second, deploying IP Multicast has an impact on the existing network infrastructure. How multicast applications and ensuing traffic will affect other services and network management remains to be seen. Third, as the Internet infrastructure and applications evolve, multicast's performance gain and deployment cost may change over time as well. New networked devices and new applications may attract more users to group applications and increase the demand for better multicast service and infrastructure support.

Fig. 1.6 illustrates the trade-off between deployability and performance over two dimensions, protocol stack and network location, of where multicast service may be implemented. At one corner, End-host Multicast is the easiest one to deploy, but suffers from performance penalty the most. At the opposite corner, IP Multicast has all the network support for the best performance, but is the most difficult one to deploy. Other designs fall in the middle. They either move up in the protocol stack or move from network core to network edges, in order to trade some performance for deployability. For example, SSM can be viewed as a service split between network layer and application layer: the network handles single-source multicast and the application handles source discovery and multi-source delivery. All previous work tried to find a fixed optimal point in Fig. 1.6 to make the best trade-off between deployability and performance. However, as we have discussed, this is a difficult design choice because the optimal point depends on applications and may drift over time. Therefore, trying to find a fixed optimal trade-off point leads to today's many incompatible multicast schemes.

1.3.2 A Ubiquitous Service with Evolvable Deployment

The idea to avoid choosing a fixed trade-off point is having a flexible design to allow the trade-off evolve over time. The evolution of deployment should take the path of least resistance, from network edges into the core. Universal Multicast (UM) is such a framework that provides ubiquitous multicast delivery and allows the implementation of multicast service evolve from one point to another in Fig. 1.6. Under this framework, multicast service can be implemented in different places, but still compatible with each other. When there is no server support or router support, applications use End-host Multicast. When such support is available, UM will automatically make use of it and applications will get better performance transparently. This approach enables ubiquitous multicast delivery immediately, thus breaking the deadlock between application development and network deployment. At the same time, it allows multicast infrastructure support gradually spread from network edges into network core driven by user demand for the service, leaving the decision of where to implement multicast functionality to end users and ISPs as part of the service's evolution. The destination of the service evolution is not assumed. The final scenario can be IP Multicast every-where, or a few Multicast Server Overlays, or even pure End-host Multicast. It will be decided by users and service providers, not protocol designers.

The adoption of World Wide Web serves as a successful example of this approach. Its basic design starts with no more than a simple client and a server, which every normal user can install and set up. As its usage increases, network operators are pushed by user demand to improve web performance by adding infrastructure supports, such as web caching, into the network. Had the original design mandated web caching for the sake of scalability, WWW would not have taken off so easily. Today, most web sites remain plain servers, while infrastructure supports are indispensable to some very popular sites, as shaped by user demand. We believe multicast can follow a similar path for its deployment on the Internet.

To achieve our goal, the design must have (1) a fully deployable starting point, and (2) a scheme that can take advantage of infrastructure support where available. Previous work fail to have both. For example, MBone facilitates transition from network edges to network core, but doesn't have a fully deployable starting point. On the contrary, End-host Multicast is deployable but does not make use of infrastructure support.

Our approach is to build an end-host overlay to connect IP-Multicast 'islands' by adaptive unicast tunnels, and use native IP Multicast within each island. The design consists of three major components: Host Multicast Tree Protocol (HMTP) for inter-island routing, Host Group Management Protocol (HGMP) for intra-island management, and a user Agent to implement the functionality on a host. It is fully distributed, self-organized, and independent from underlying routing protocols. We evaluated the system performance through simulation, implemented a prototype and tested with existing multicast applications.

1.4 Dissertation Outline

The rest of this dissertation is structured as follows. In Chapter 2, we present design rationales, overview of UM architecture and the system components. Chapter 3 discusses the design of HMTP, a simple and scalable End-host Multicast protocol, and its performance evaluation through simulations. In chapter 4, we discuss HGMP design from simple single-DM model to a more complex multi-DM model, and how it is flexible enough to accommodate different multicast schemes. In chapter 5, we discuss issues in implementation of the UM Agent, including design choices and tests with applications. We discuss related work in chapter 6 and conclude in chapter 7.

CHAPTER 2

Overview

In this chapter, we explain the rationales behind the Universal Multicast design, and give an overview of the system architecture and components.

2.1 Design Rationales

2.1.1 The Goal

The goal of Universal Multicast is to provide *ubiquitous* multicast delivery service that allows infrastructure support to *evolve* from network edges to network core. In the protocol stack, UM forms a sub-layer between application layer and transport layer. Like IP Multicast, it provides only best-effort multicast packet delivery service on top of UDP; functionalities like reliable delivery, security, and group management are left to upper layer protocols or applications to implement. The interface between UM and application layer should be well-defined and relatively fixed; the interface between UM and transport layer should be flexible to accommodate various network environments. Therefore UM provides consistent multicast service semantics and API to upper layer applications across networks with different degrees of multicast support. Infrastructure support for multicast can be deployed gradually by local network administrators and service providers to improve the quality of service, but the type, scale and place of such support are hidden by the UM sub-layer from applications.

2.1.2 The Requirements

In order to provide ubiquitous and evolvable multicast delivery service, the UM design should meet two fundamental requirements:

1. *Have a deployable and efficient starting point*

2. *Allow service evolution by being open to infrastructure support.*

A good staring point means even when there is no network support at all, UM should still be *deployable* and adequately *scalable* for normal applications. This is necessary for applications to be successful at the initial stage, making it possible for the usage to reach critical mass and spur infrastructure support. The repetitive unicast is not suitable because its scalability problem limits it to small groups only. An End-host Multicast scheme fits the requirement the best. It is easy to deploy since users only need to install a software on their own computers at their own will, and it does not require any support from routers, servers, operating systems or administrative efforts from service providers. It has better scalability since all group members share the workload.

There are three points of being open to infrastructure support. First, UM should keep the IP Multicast service model to the furthest extent, thus being *compatible* with IP Multicast, the dominant router-based multicast scheme on the current Internet. UM can leverage IP Multicast's installation base and past research efforts to benefit applications. For example, The performance of packet delivery will be improved in IP-Multicast-enabled networks, and modifications to existing multicast applications would be minimal. When there are different infrastructure multicast support, a single backward-compatible service model gives applications consistent service semantics and API, which is important to hide lower layer's heterogeneity from the application layer.

Second, UM should be *flexible* to accommodate various network environments, including different End-host Multicast schemes and different kinds of infrastructure support, such as local proxy, server overlay, and native multicast routers. Therefore, UM is able to not only make use of existing multicast schemes, but also take advantage of future progresses. As a general framework, Universal Multicast is not bound to any particular routing protocol of either IP Multicast or End-host Multicast.

Third, the system should be fully *self-organized* in order not to compromise its deployability. A self-organized system should be able to automatically detect and make use of network support where available, without requiring any external help such as administrative coordination. Using external help would make the system design easier, but at the expense of deployability. Multicast Server Overlay is an example of relying on dedicated servers to improve performance and facilitate management, but sacrificing deployability.

2.2 Architecture and Components

Like IP Multicast, Universal Multicast is not a single protocol but a set of components at various levels (Table 2.1). One key difference between IP Multicast and Universal Multicast is that the former's network infrastructure is set up and managed by administrators, but the latter's is fully self-organized by end hosts.

Fig. 2.1 shows a sample UM group illustrating UM's architecture. In UM's view, the Internet is composed of many IP-Multicast-enabled *'islands'*, which are separated by unicast-only routers. Every member host runs a daemon program

	IP Multicast	Universal Multicast
Host Extension	OS kernel support	A user-level Agent
Local Membership	IGMP	HGMP
Intra domain/island Multicast Routing	DVMRP, MOSPF, PIM, CBT	Using deployed IP Multicast
Inter domain/island Multicast Routing	MASC/BGMP, MBGP/MSDP/PIM	End-host Multicast protocols (e.g., HMTP)

Table 2.1: Analogy between IP Multicast and Universal Multicast

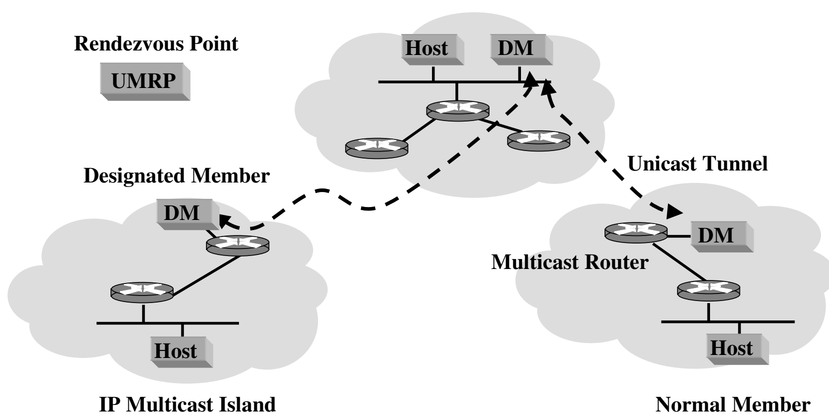


Figure 2.1: Universal Multicast architecture

(UM Agent) in user space to provide the UM functionality. For each group, one or more group members in each island are elected as Designated Members (DM), which build unicast tunnels connecting all islands. Application data are encapsulated in UDP packets flowing through tunnels from one island to another , but are delivered via native IP Multicast within every island.

Island An island is a network of any size that supports IP Multicast. It can be as large as a network with multiple ASs (e.g., MBone), and as small as a single host. An island's boundary does not need explicit configuration; it is just the furthest extent that an IP Multicast packet can reach in the network. If operators enable more IP Multicast routers, the island will automatically expand accordingly as IP Multicast packets will reach wider area. When there is no IP Multicast support at all, a single host itself is considered as an island, since most modern operating systems support IP Multicast. For each multicast group, each participating island (i.e., islands with at least one group member in it) has an identifier called *island number*, which is unique across all participating islands of the group. One island usually have many island numbers, each for one multicast group, and these island numbers are totally independent from each other. The assignment and maintenance of island number are described in Chapter 4.

Group Member UM adopts the same open group model as IP Multicast's. Hosts that are willing to receive group traffic join the group and become group members. Senders are not required to join the group. Therefore, UM supports non-member senders, who only want to send packets to the group, but not receive any group traffic. For each group, one or more members will be elected as Designated Members (DM) in each participating island. A DM's main tasks are running an End-host Multicast protocol to build inter-island tunnels with DMs in other islands, and forwarding packets in and out of the island. When there are multiple DMs in the same island, one of them is elected as the *Head DM*, and the rest are *Tail DM*s. In this dissertation, we present an End-host Multicast protocol called HMTP (Chapter 3) for inter-island routing, and an intra-island protocol called HGMP (Chapter 4) for DM election and coordination.

Agent Every UM host runs a daemon program called UM Agent, which implements HMTP, HGMP and other components to provide full UM functionality. The Agent is implemented in user space to avoid changes to operating system. Applications communicate with the Agent via a UM library to join and leave groups. Details of the Agent implementation is described in Chapter 5.

Tunnel Different islands are connected by unicast tunnels built between their DMs. Unlike MBone's IP-in-IP tunnel, the UM tunnel encapsulate data packets in normal UDP packets to avoid changes to operating system, and carry more information.

Rendezvous Point (UMRP) One fundamental problem in multicast routing is how senders and receivers get to know each other. In most End-host Multicast protocols, this problem is solved by using a rendezvous point (RP). RP maintains some information about the current overlay, and new members contact RP to find existing members. We adopt the same approach in UM design. In UM, each group has a rendezvous point, called Universal Multicast Rendezvous Point (UMRP). Its main purpose is to help a new DM join the inter-island overlay. UMRP keeps information of which End-host Multicast protocol the group uses, and corresponding protocol-specific information, e.g., the root of the shared-tree for HMTP. A new DM queries the UMRP first, knowing the routing protocol and its parameters, then run the protocol to start joining the overlay. Unlike the rendezvous in PIM or the core in CBT, UMRP is not involved in packet forwarding, so its location does not have significant impact on the performance of data delivery. One UMRP can serve multiple groups. It can be a normal end host, a dedicated server, or a cluster of servers. Failure of UMRP will prevent new DMs from joining the inter-island overlay, but will not affect data dissemination among existing islands.

Group Identifier (GID) Globally unique identification of a group is important to Universal Multicast. If two groups created in two separated islands have the same address, when the two islands are connected by a unicast tunnel, members of one group will receive the other group's traffic. It will consume bandwidth and confuse applications. Since the current Internet does not have a global multicast address allocation architecture, we cannot simply use class D addresses. Instead, UM uses the combination of UMRP's IP address and a 32-bit *group number* as a unique Group Identifier (*GID*). The group number is assigned by UMRP upon the creation of the group and is unique among all groups served by the same UMRP. Therefore, GID can uniquely identify a group globally. The assignment of group number is a local decision made by UMRP, without any multi-host coordination required. A user-friendly *group name* may be used in place of the group number. UMRP will be responsible for resolving group name to group number. The GID, or UMRP and group name, is obtained off-line by users or applications, and passed to the UM Agent to join the group. For example, to join a UM group called *forest.cs.ucla.edu/mytalk*, application will use DNS to resolve the UMRP *forest.cs.ucla.edu* to *131.179.96.162*, and then contact the UMRP to resolve the group name *mytalk* to a group number *1234*. The GID now is *131.179.96.162/1234*. It is then used in both data packet and routing packet header to identify the group.

Local Multicast Groups Though GID uniquely identifies a UM group, only class D addresses can applications, operating systems and routers understand. Since intra-island packet delivery is done by IP Multicast, local IP Multicast groups must be used within every island. For each UM group, every island with group members in it will have three local IP Multicast groups: *DATA_ GROUP* for transmitting application data packets, *ASSERTION_ GROUP* for electing DMs, and *DM_ GROUP* for coordinating multiple DMs. These local groups are created by the Head DM in each island. The group addresses are obtained by any allocation scheme available in the island. Local multicast groups of one island are totally independent from those of another island.

Group Directory For each UM group, its GID is globally unique, but its local IP Multicast groups are independent from each other across different islands. Therefore, in every island, applications need to be able to find out local groups given the GID, and vise versa. This mapping between GID and local groups is done by the Group Directory. In each island, after creating local groups, the Head DM announces the mapping to a reserved IP Multicast group, *GD_ GROUP*. GD_GROUP has a fixed, well-known class D address across all islands. Other group members listen to GD_GROUP and collect the mapping information. Group Directory works in a similar way to multicast session directory, but with improvements like fast start-up.

Data Flow In UM, applications send and receive native IP Multicast packets via the DATA_GROUP. When a packet is sent out by the application, member hosts (and applications running on them) in the same island receive it via IP

Multicast. The group's DM in this island will append an inter-island header to the packet, and forward it to neighbor islands via tunnels according to the inter-island routing protocol. Upon receiving a tunneled packet, DM replaces the encapsulation header with an intra-island header and send it to local island by IP Multicast. Agent on each member receives this packet, strips off the header, and feeds the payload to its applications via the loop-back interface.

CHAPTER 3

Host Multicast Tree Protocol (HMTP)

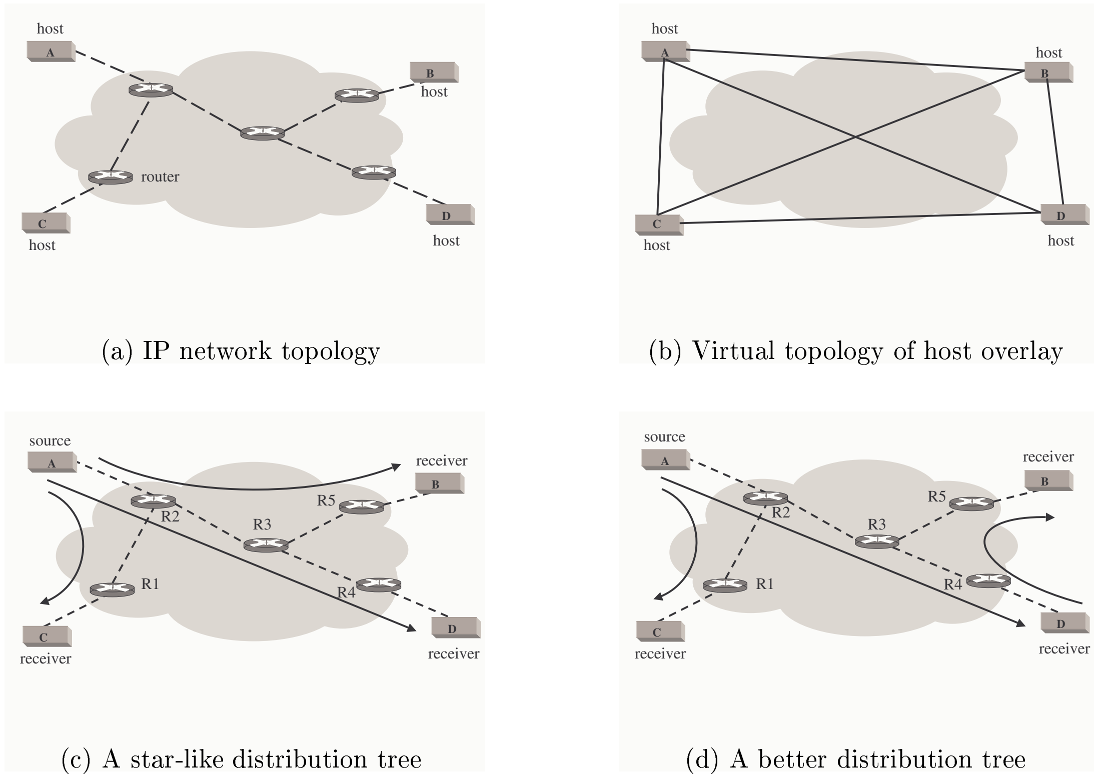
3.1 Multicast Routing on End-host Overlay

3.1.1 The Problem

Overlay networks composed of end-hosts are different from IP networks. In IP networks, a router has multiple but a fixed number of physical network interfaces, with each connecting to a direct neighbor router. Therefore the topology is a connected graph with connectivity greater than a tree but less than a full-mesh (Fig. 3.1a). IP Multicast routing protocols build shortest paths from receivers to the source or the rendezvous point for data delivery. However these protocols cannot be used for multicasting on end-host overlay.

An end-host usually has only one physical network interface. But in the overlay, it has virtual links directly to every other host because of the end-to-end connectivity provided by underlying IP network. Thus the resulting end-host overlay topology is a full-mesh (Fig. 3.1b). Applying IP Multicast routing algorithms to a full-mesh will always produce a star-like distribution tree (Fig. 3.1c) where every receiver is directly connected to the source. This is the same as using unicast and has serious scalability problem and robustness problem near the source.

Therefore we need an end-host multicast protocol which can produce a better



(a) IP network topology

(b) Virtual topology of host overlay

(c) A star-like distribution tree

(d) A better distribution tree

Figure 3.1: Multicast routing on end-host overlay

distribution tree out of the full-mesh substrate. Fig. 3.1d shows an example of a better distribution tree, in which one receiver (D) forwards the packet to another receiver (C). The result is that member hosts share the workload of forwarding packets, but some hosts will experience longer latency of packet delivery. A distribution tree congruent to the underlying IP network topology has less delay penalty and makes the forwarding more efficient. However, the underlying network topology is unknown to end-hosts. Therefore, an end-host multicast protocol needs to do end-to-end measurements to infer some information about the IP network topology and builds distribution tree accordingly.

3.1.2 Approaches

There have been several approaches proposed to tackle the problem of building multicast distribution trees in end-host overlay. The *centralized* approach depends on a controller to direct members to do measurements, collect measurement results, compute a minimum spanning tree among members and disseminate routing tables. Though it has the potential to compute other types of distribution trees, the centralized nature limits its scalability.

In the *mesh-based* approach, the full-mesh overlay topology is first reduced to a mesh topology, then a conventional IP Multicast routing algorithm is applied to generate the distribution tree. The challenging part of this approach is in determining which virtual link to keep, which to delete, and how to do both in an adaptive way. The advantages of this approach are that existing multicast routing algorithms can be leveraged, and loop avoidance is built-in. Unfortunately, current protocols following this approach requires every member to keep a full list of all the other members. The needs for member discovery and maintenance of membership list limit these protocols to support only small multicast groups.

In the *tree-based* approach, member hosts self-organize into a tree structure by explicitly picking a parent for each new member. This approach scales well to large group sizes because each host only needs to know of a small number of other hosts. However, the choice of parents determine the performance, and to a large degree the efficiency, of the resulting tree. It must also prevent loops and handle tree partition.

There are also some other approaches. For example, CAN organizes member hosts into an n-dimensional virtual space and forwards packets based on the incoming dimension number. NICE organizes member hosts into a multi-layer hierarchy and packets are forwarded based on the incoming layer number.

Factors that affect the evaluation of a protocol include end-to-end delay, tree cost, scalability, robustness and protocol complexity etc. Protocols may make different trade-offs to meet different application needs, especially when the protocol is incorporated within applications. HMTP is a simple and scalable tree-based protocol that provides end-host multicast functionality with low tree cost and moderate end-to-end delay. As more end-host multicast protocols and applications are emerging, we expect multiple protocols will co-exist in the future, and Universal Multicast framework has been designed to be protocol-independent.

3.2 HMTP [1]

3.2.1 Rationales

HMTP is a tree-based end-host multicast protocol. It builds a group-shared tree instead of a source-specific tree. Routing states associated with source-specific tree grow linearly with the number of data sources. While the impact of this linear growth on end-hosts is less of a concern than on routers, it is still beneficial to limit it. Hence our choice of group-shared tree over source-specific tree. Source-specific trees are known to have latency advantage over group-shared trees; however, since end-host multicast trees are built on an overlay network that does not necessarily reflect the underlying physical connectivity of the network, we decided that performance advantage of source-specific trees is not as significant in end-host multicast as in IP Multicast.

To reduce routing inefficiency, an overlay multicast tree should be congruent to the underlying network topology to the furthest extent possible. In HMTP,

[1] **In Universal Multicast, only Designated Members participate in tree construction and maintenance. In this section, we use "member" to mean "Designated Member" for ease of exposition.**

however, we assume that members have no knowledge of the underlying physical network. While tools such as `traceroute` are available for discovering such information, they are usually not very dependable because intermediate routers could block their probes. Furthermore, running such tools prior to data delivery may take longer than the data delivery time itself. Hence without knowing the underlying physical network topology, end-hosts running HMTP use end-to-end measurements of some network properties to serve as distance metric. Currently HMTP uses member-to-member round-trip time (rtt) as the only distance metric in tree building. In the future we may add bottleneck bandwidth as a second metric.

A tree is more fragile than a mesh in that a single node failure or a loop can partition the tree and disable communications among members. In HMTP, members maintain the tree structure by periodically exchanging messages with neighbors. The mechanisms to detect node failure and loop formation are described below.

3.2.2 Join

So as to reduce the total cost of the tree and to maintain a maximum degree constraint at every host, HMTP tries to cluster nearby members together. The simplest way to achieve such clustering would be to let each new member choose as its parent an existing member closest to it. This simple scheme, however, requires someone to maintain a list of all group members. Such a list is not necessary if each new member chooses as its parent only the closest from a random subset of existing members. A tree so generated, however, will have a very random (and most likely grossly suboptimal) structure. In HMTP we define some rules to generate a partial list of existing members. The rules ensure that when a new

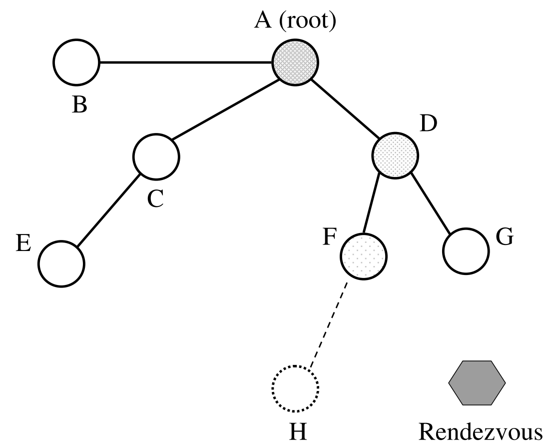


Figure 3.2: H joins a tree consisting of A through G

member joins the tree by choosing the closest member from this list as its parent, the resulting tree will not be totally random.

In Fig. 3.2, node H is a newcomer joining a group. It knows of the group's UMRP from the group's GID. By querying the UMRP, it learns that node A is the root of the tree. H sets A as its potential parent and asks A for a list of A's children. From the list of A and A's children, H picks the closest one, in this case D, as a new potential parent. H repeats this process and finds the next potential parent, F. In the next iteration, H finds that the new potential parent remains F. So H attempts to make F its parent by sending a join request to F. It is the potential parent who decides whether to accept a join request, based on its policy, bandwidth, traffic load, etc. If F rejects H's request, H marks F as invalid and goes back up one level and resumes the search for a parent. It eventually finds G. The detail algorithm is described in Fig. 3.3 .

To summarize, a newcomer tries to find a good parent by searching a small part of the tree. It stops when it reaches a leaf node, or a node that is closer

1. All members are regarded as valid potential parents by default. A stack S is allocated for storing past potential parents.
2. Find the root by querying UMRP. Set the root as potential parent.
3. Query the potential parent to discover all its children. Measure rtts to the potential parent and its children.
4. Find the nearest member among the potential parent and all its children, except those marked as invalid. If all of them are invalid, pop the top element in S and make it the new potential parent, return to step 3.
5. If the nearest member is not the current potential parent, push current potential parent onto the stack, set the nearest member as the new potential parent, return to step 3.
6. Otherwise, send the potential parent a join request. If refused, mark the potential parent invalid and return to step 4. Otherwise parent found, establish a unicast tunnel to it.

Figure 3.3: Join Algorithm

than all its neighbors. The algorithm scales to the number of group members. It does not guarantee that the parent chosen is the nearest one among all members; however, since all members follow the same rule, the distance information is encoded into the tree structure and should help every newcomer. For example, two nearby hosts will have similar delay to other hosts, so they are likely to make the same decision at each step. When one of them joins the tree, it may not choose the nearest existing member. But when the second one joins, it probably will choose the first one as its parent. Therefore nearby members are more likely to be clustered together.

1. Push root path into the stack S. Delete self and current parent (which is leaving or has crashed) from the top of the stack.

2. Pop grandparent as potential parent from the stack, wait for a random delay, then start running the join algorithm from step 3, until a new parent is found. The random delay is used to prevent all orphans from contacting the grandparent at the same time.

Figure 3.4: Repair Algorithm

Clustering nearby members makes the tree structure congruent to the network topology to the first order. Links with large latency will be traversed fewer times. Members behind such links (e.g., members on dial-up lines) are likely to be pushed to the edges of the tree. Given the same set of members but different join sequences, the protocol will produce different trees. However, with periodic tree improvement algorithm (see Section 3.2.5) run by every member, these trees will eventually converge to some stable structures that have similar gross qualities. For example, if a group has a dial-up host with large last-hop delay, and some other well-connected hosts with relatively short delay, the well-connected hosts should form the core of the tree and the dial-up host will connect to the core by a single link after the tree stabilizes, regardless of the members' join order. Even if the dial-up host is the first member to join the tree and becomes the root, the shape of the resulting tree should be similar with similar gross quality. This is confirmed by our simulations with random join sequences.

3.2.3 Maintenance

States in HMTP are refreshed by periodic message exchanges between neighbors. Every child sends REFRESH messages to its parent. The parent replies by sending back PATH messages. PATH messages sent by the parent contains the root path of the parent. By appending itself to its parent's root path, a member constructs its own root path. Every member must maintain the freshness of its list of children and its root path. The root sends REFRESH messages to UMRP, so that UMRP always knows who is the current root.

3.2.4 Leave

When a member leaves a group, it notifies its parent and children. Its parent simply deletes the leaving member from its children list. It is the leaving member's children's responsibility to find new parents. A child looks for a new parent by running the join algorithm in reverse order (Fig. 3.4). If the root is leaving, its children contact UMRP after a random delay. The first member to contact UMRP becomes the new root.

3.2.5 Improvement

As network condition and group membership change over time, members may need to restructure the tree, by switching to new parents, to improve performance. Parent switching in HMTP is done by periodically re-running the join procedure (Fig. 3.3). To reduce not only the workload of members near the root, but also the time needed to complete tree restructuring, members do not start the re-joining procedure from the root, but from a randomly-picked node in their root paths. Furthermore, in step 4 of the re-join procedure, a member other than the closest one could be picked as the next potential parent. This will allow members to search other branches of the tree for better parent.

A member may switch to a new parent if the new parent is closer than the current one. To avoid oscillation, we enforce a threshold on delay gain in triggering a switch. Another reason for enforcing a delay gain threshold in parent switching is that even though when a member switches to a closer parent it benefits all of its descendants, it also invalidates its descendants' root paths. To update its descendants' root paths after a switch, a member can optionally push a message down its subtree, instead of waiting for regularly scheduled PATH messages.

Tree improvement runs less frequently than the sending of REFRESH and PATH messages. An interval of a couple of minutes or more may be sufficient. The actual frequency varies for each member, depending on how far away it is from its current parent. A member that already has a close parent does not need to run tree improvement frequently. Considering that most dial-up hosts have long last-hop delay, we decided to remove the last hop delay in evaluating a member's distance to its parent. Note that hosts connected to a LAN usually has negligible last hop delay in the first place. In most networks, the last hop delay can be obtained by pinging the default router.

3.2.6 Partition Recovery

When a non-leaf member crashes, the tree is partitioned. Surviving members must be able to detect the failure and repair the tree. Repairing the tree upon a member crash is similar to handling a member leave. The difference is that surviving members usually do not receive prior notification of a crash. Hence node failure is detected by noticing repeatedly missing REFRESH or PATH messages. When a node failure is detected, the parent of the failed node simply updates its children list; the children of the failed node must run the repair algorithm. As long as a node on its root path or the UMRP is available, a partitioned member can always rejoin the tree. When Universal Multicast becomes widely deployed, most groups will use a dedicated server as UMRP, which will increase robustness.

In the rare cases when all hosts in a member's root path and the UMRP crash at the same time, the best thing this member can do is to try connecting to other group members in its cache. In the process of joining the tree or during tree improvement, a member learns of members in other branches of the delivery tree. These members can be cached and used for partition recovery. Members with short root path may want to cache more such members. Using cached members does not, however, guarantee that the tree can be reconnected. In the worst case, the group could be partitioned into pieces that cannot find each other. Existing end-host based overlay networks usually require every member to maintain an accurate and complete member list to guarantee partition recovery. Such a requirement limits the scalability of these protocols. HMTP achieves partition recovery as long as there is a surviving host in a member's root path or the UMRP is not down.

3.2.7 Loop Detection and Resolution

If a member switches to one of its descendants, a routing loop is formed. Members in a loop will see itself in its root path. To prevent loop formation, a member checks that it is not in its potential parent's root path before settling on a new parent. With this simple algorithm, loop can still be formed if there are concurrent topology changes, e.g., in Fig. 3.2, if C joins F and D joins E at the same time, a loop will form. Forming a loop depends on the timing of multiple parent switches. In HMTP, we use loop detection and resolution instead of loop avoidance to handle routing loops. When a loop is formed, a member in the loop will detect the loop after its root path is updated. Upon detection of a loop, the member stops passing its root path downstream, breaks the loop by leaving its current parent, and re-joins the tree from the root. In a tree structure the existence of a loop also means a tree partition. Hence the member detecting a loop must immediately break the loop and re-join the tree. If multiple members detect the loop at the same time, the loop will be broken into multiple pieces. Each piece is loop-free and will be re-joined to the tree independently.

3.2.8 Non-Member Senders

IP Multicast service model allows non-member senders. That is, a host can send packets to a group without joining the group. To be fully compatible with IP Multicast service model, HMTP needs to support non-member senders too. A non-member sender runs the join algorithm to find a parent on the HMTP tree as a normal member does. When it connects to the parent, this link is labeled as "sender-only", which means the parent will not forward any packet to to the link, nor report it in children list to others. Therefore, the non-member sender will be attached to the tree always as a leaf node and will not receive any data packets.

3.3 Enhancements

The basic tree algorithm is simple and scalable, but some extra work is needed to improve its performance.

3.3.1 Delay Measurement

HMTP uses delay as the routing metric. If Internet distance services like IDMaps [FJJ01] are available, we can use them to efficiently determine latencies between hosts. When such services are not available, end hosts must measure inter-host latencies themselves, by using `ping` or equivalent tools, for example. To reduce measurement overhead, results can be *cached*. Another way to reduce measurement traffic is to use a synchronized *group clock* at every host to enable one-way measurement. A group clock keeps track of milliseconds elapsed since the inception of the group. Every member synchronizes its group clock to its parent's by piggy-backing time-stamps in REFRESH and PATH messages; eventually every member's group clock is roughly synchronized with the UMRP's. Even without actual measurements, either one-way or round-trip, a host can estimate link delay from existing information, e.g., for three hosts A, B and C, the delay $D_{A,B}$ can be estimated by triangulation:

$$|D_{A,C} - D_{B,C}| < D_{A,B} < D_{A,C} + D_{B,C}$$

assuming shortest path unicast routing [FJJ01].

3.3.2 Join Delay and Foster Care

One problem of the basic join algorithm is long join latency, especially for large group size and sparse group. To reduce join latency, we allow new members to temporarily attach to a random member. Existing members must accept a limited number of temporary (foster) children for a short period of time. After that time, the parent can remove the foster children. Data packets are forwarded to all of a node's children, including its foster children. However, a node does not list its foster children in reply to a join query (step 3 of Fig. 3.3).

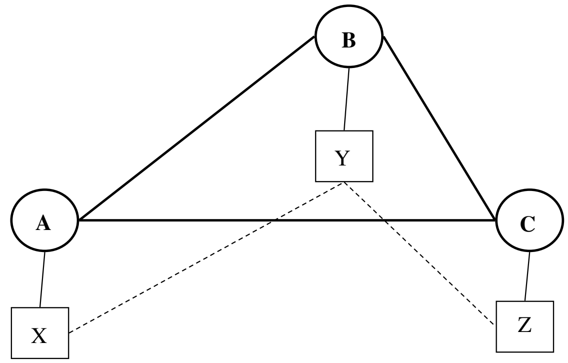


Figure 3.5: X, Y and Z are routers; A, B and C are end hosts. $D_{A,C} > D_{A,B} > D_{B,C}$

3.3.3 U-Turn and Triangle Optimization

Suppose there are three hosts A, B, and C, attached to three routers X, Y, and Z respectively (Fig. 3.5), and the delays between A, B, and C are $D_{A,C} > D_{A,B} > D_{B,C}$. When constructing a spanning tree among these three hosts, we prefer to discard the longest virtual link AC. However, since end hosts do not have knowledge about the router topology (e.g., there is no physical link between X and Z), they may make wrong decisions; for example, suppose A initiates a new multicast group. It is the first member of the group and becomes the tree's root. If C joins before B, the tree becomes A-C-B. Packets from A go to C first, then are forwarded back to B. We call this the *U-turn* problem. Since B is currently a child of C, C will never use B as a potential parent. And since the B-C distance is smaller than the B-A distance, B will always pick C as parent. So tree improvement cannot solve this U-turn problem.

Our solution is to give B more information to make the correct decision when it joins the tree. When A passes its children list to B, it also includes the delay from A to all its children (in this case, $D_{A,C}$). B measures the delay $D_{A,B}$ and $D_{B,C}$

following the join algorithm. Now B knows all three link delays. It will choose A as its parent unless $D_{A,B}$ is the longest one among the three virtual links. After B joins A, C will discover B during tree improvement and switch to B. We call this *triangle optimization*. A node's latencies to all of its children are always available because of the periodic exchanges of REFRESH and PATH messages. In the case that the potential parent has more than one existing children, the newcomer shall apply triangle optimization between the potential parent and the child who is the nearest to the newcomer.

3.3.4 Server Backbone

Using end users' computers to forward packets makes the design deployable. However, for applications that service performance and stability are more critical, there is a need of using a server backbone to provide the multicast forwarding service. In this scheme, dedicated servers are strategically deployed on the Internet by a third-party such as a CDN (content distribution network) company. These servers run an end-host multicast protocol among themselves to build a multicast overlay, which can receive content streams from content providers and deliver them to end users. Scattercast [Cha00] and AMcast [STW01] take this approach of multicast server overlay as an essential part in their design because dedicated servers usually are more stable and have more bandwidth and computing power than end user's computers.

UM does not require the use of server overlay, but it can easily incorporate it if there are servers available. One approach is to associate each host with a *rank* when it joins the HMTP tree. A node will only join a parent who has higher or equal rank. A parent can ask a child to leave if there is a need to make room for another child with higher rank than the old child's. End users' computers have default low rank, e.g, below 10. Servers can be configured with higher rank, e.g., above 10. Security measures may be needed to prevent false claim of one's rank. But it is up to applications and is orthogonal to UM design. After joining the HMTP tree, dedicated servers will occupy the top of the tree, while regular hosts are pushed down to the edge. Thus we will have a stable server backbone as the core to serve the group.

3.4 Performance Evaluation

3.4.1 Performance Metrics

We conducted simulations to evaluate the performance of HMTP against Unicast Star and IP Multicast, including shortest path source tree (SPST) and shortest path group tree (SPGT[2]). The quality of a tree is judged by the following metrics [WE94]:

- Tree Cost: the cost of a tree is the sum of delays on the tree's links. Tree cost is a convenient, though somewhat simplified, metric to capture total network resource consumption of a tree. The ratio of a tree's cost to that of a corresponding SPST is the tree's *cost ratio.*

- Tree Delay: delay from one member to another along the tree. The ratio between tree delay and unicast shortest path delay is *delay ratio. Group diameter* is the maximum delay between any pair of members. It represents the time after which a packet is assured to reach every member. The ratio of group diameter to the diameter of an SPST is the *group diameter inflation.*

- Link Load: the link load of a physical link is the number of duplicates the

[2] We use bi-directional shared tree as in Core Based Tree.

link has to carry when a packet is multicast to the group.

Results presented here are based on simulations on a network consisting of 1000 nodes, representing routers, and 3300 links. The network has a random flat topology generated using the Waxman model [ZCB96]. We generated some additional nodes to serve as end hosts and randomly attached each of these additional nodes to a router node. The connection between an end-host node and the router node it attaches to is small but not negligible (see Fig. 3.8). The maximum node degree constraint when running HMTP is set to eight. Except for some results from a single run (Figs. 3.6, 3.10, and 3.15), data points in the following graphs represent averages over 100 runs with 95% confidence interval. At the end of this section, we also present results from an experiment on a snapshot of real Internet topology.

3.4.2 Simulation Results and Analysis

3.4.2.1 Convergence

Fig. 3.6 shows the result from a typical run to construct an end-host multicast tree with 100 members. Members join the group one by one within the first minute of simulated time, causing the sharp initial increase in tree cost ratio (y-axis). In the simulations, members run the tree improvement algorithm once every 30 seconds. By the second minute, tree cost ratio has decreased to a stable point. In most of our simulations, the tree stabilizes and there is no more changes to the tree structure after four to five runs of tree improvement algorithm by each member. Note that even without running the tree improvement algorithm, the tree cost ratio is relatively low at peak (about 1.5 times that of SPST). We attribute this to how the join algorithm itself helps newcomers find a close by

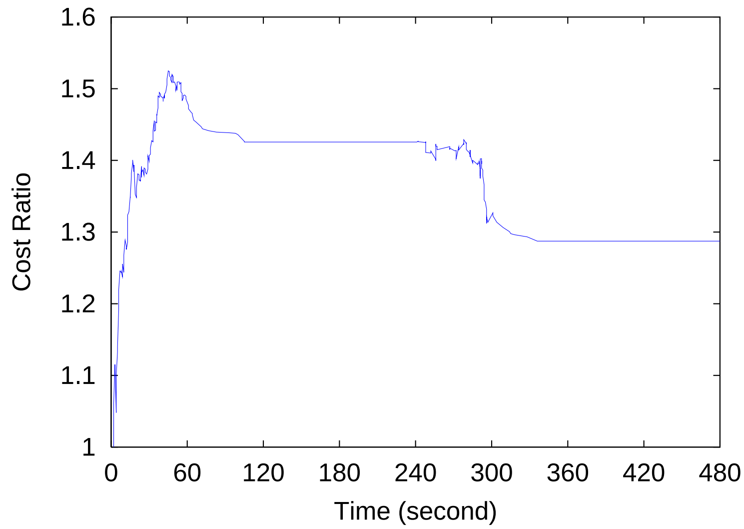


Figure 3.6: HMTP Tree Cost Ratio vs. Time, with members join and leave.

parent. In the simulated scenario, fifty members leave the tree in random order during the fifth minute of simulated time. After the fifty departures, it takes the remaining members less than a minute to settle upon another stable tree. We conclude that HMTP tree can converge quickly in the face of drastic group membership changes.

3.4.2.2 Tree Cost

We next experimented with various multicast delivery trees connecting 20 to 500 members to compare their tree costs. For each delivery tree, we randomly pick a member to serve as the source (for SPST and Unicast Star) or rendezvous host (for SPGT) or root (for HMTP), and calculate the tree cost of the resulting tree. As expected, SPST and SPGT have cost ratio of 1, while HMTP and Unicast Star have larger cost ratio (Fig. 3.7). Also apparent from the figure is that HMTP's cost increases very slowly with larger group sizes. More interestingly, we show in

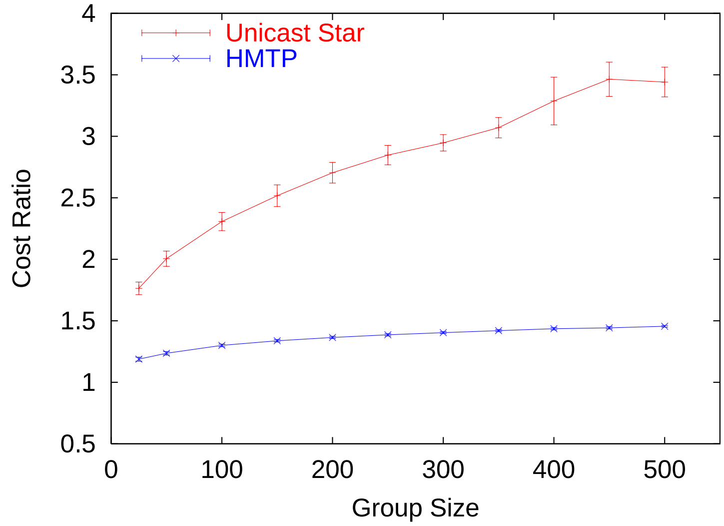


Figure 3.7: Cost Ratio of HMTP and Unicast Star. Last hop delay is 3ms.

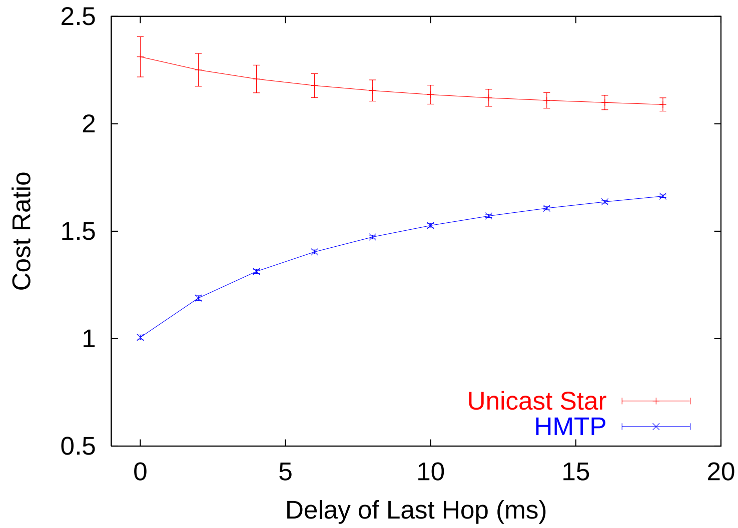


Figure 3.8: Tree Cost v.s Last hop delay with 100 members.

Fig. 3.8 that while larger last-hop delays lead to higher cost ratio on HMTP, the opposite is true for Unicast Star. In order to analyze this behavior more closely, we next split tree cost into two components: C_{last}, to account for the cost of all last hop links on the tree, and C_{net}, to account for the cost of all internal network links. The cost of SPST, Unicast Star, and HMTP trees can be expressed as follows using these two components:

$C_{spst} = C_{spst\_last} + C_{spst\_net} = N*D_{last} + C_{spst\_net}$,

$C_{star} = C_{star\_last} + C_{star\_net} = 2(N\text{-}1)*D_{last} + C_{star\_net}$,

$C_{hmtp} = C_{hmtp\_last} + C_{hmtp\_net} = 2(N\text{-}1)*D_{last} + C_{hmtp\_net}$,

where D_{last} is the last hop delay and N is the number of group members. For both Unicast Star and HMTP, the ratio of the multiplier in the C_{last} term over that of SPST is (2 - $\frac{2}{N}\approx$ 2). While for the multiplier in the C_{net} term (which is not expanded out), the ratio over that of SPST is 2.3 for Unicast Star and 1 for HMTP (see Fig. 3.8 when last hop delay is zero). Hence as D_{last} increases, the ratio of a tree's C_{last} over that of SPST contributes more to the overall cost ratio, which explains why Unicast Star's cost ratio decreases while that of HMTP increases. Compared to IP Multicast, end-host-based multicast introduces transmission overhead at two places: network core and last hops. The above analysis shows that in terms of tree cost, all end-host based multicast protocols have the same overhead attributable to last hop connections. The difference in their tree costs lies in the overhead attributed to routing on overlay topology, reflected by the ratio of their C_{net} over that of SPST. This ratio can be used as a benchmark to compare different end-host based multicast routing protocols. HMTP's value for this ratio is fairly low (below 1.1) and remains flat as group size increases (Fig. 3.9). While Unicast Star's value for this ratio is big because it incurs many duplicate packets in internal network links, which also increases with larger group

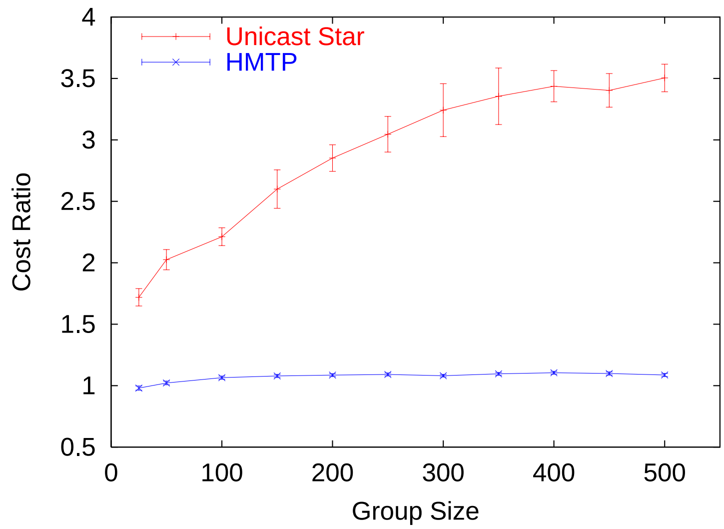


Figure 3.9: Network Cost Ratio of HMTP and Unicast Star vs. Group size. Last hop delay is 0.

sizes. Another disadvantage of Unicast Star is that its cost ratio has a wide distribution. Very large cost ratio is observed when the source is behind a long distance link. HMTP avoids this by letting other members, not just the source, forward data packets.

3.4.2.3 Tree Delay

Group shared trees such as SPGT and HMTP incur penalty on end-to-end delay. Fig. 3.10 shows the probability density function of delay ratio among all pairs of members in an HMTP tree. Most pairs have delay ratio less than 5, but the worst one reaches 12. Note however, high delay ratio does not necessarily mean large absolute delay. Large absolute delay is reflected in large group diameter inflation. Fig. 3.11 shows the average and 90%-tile delay ratio from our simulations as we increase the group size. The corresponding group diameter inflation is shown

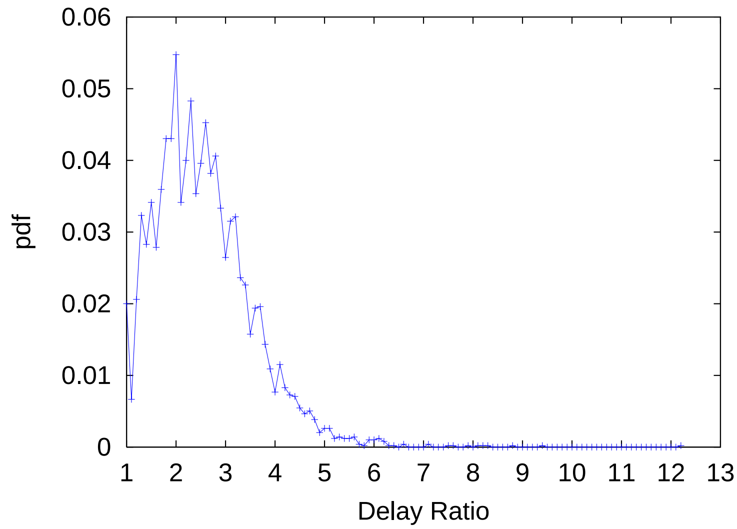


Figure 3.10: Distribution of delay ratio in a HMTP tree of 100 members. Last hop delay is 3ms.

in Fig. 3.12. As delay ratio increases in the former graph, the group diameter inflation stays largely flat in the latter graph. We again apply similar separation of delay overhead into network core component and last hop component. Fig. 3.13 shows overall tree delay ratio as we increase the last hop delays. Let D_{net} be the delay from the source to a destination, excluding both last hops. Then the delay of a single path can be expressed as:

$$D_{unicast} = D_{unicast\_last} + D_{unicast\_net} = 2D_{last} + D_{unicast\_net},$$

$$D_{spgt} = D_{spgt\_last} + D_{spgt\_net} = 2D_{last} + D_{spgt\_net},$$

$$D_{hmtp} = D_{hmtp\_last} + D_{hmtp\_net} = (2 + 2H)*D_{last} + D_{hmtp\_net},$$

where H is the number of intermediate end-hosts between the source and destination (on an end-host based multicast tree). The ratio over unicast of the first term is 1 for SPGT, and (1+H) for HMTP. The mean of the ratio over unicast of the second term is 1.7 for SPGT, and 2.3 for HMTP (see Fig. 3.13 when last hop delay is zero). As D_{last} increases, the first term contributes more to the overall delay ratio. Hence SPGT's delay ratio decreases towards 1. Since H is likely to be larger than 1.3, HMTP's delay ratio increases. Unlike the analysis for tree cost where the contribution of the first term depends only on last hop delay, the first term in the tree delay expression for HMTP depends also on the number of hops traversed, which is determined by the end-host multicast routing algorithm. This means that the delay overhead of HMTP cannot be attributed to the second term alone. Fig. 3.14 shows that delay ratios stay largely flat as we increase group size beyond 100 members when the last hop delay is zero. A more compact tree structure with smaller H will result in smaller delay ratio.

Increased tree cost and end-to-end delay over native multicast is expected for all end-host based multicast delivery trees. Delay results presented here include delays between all pairs of members in a multicast group. In reality, not all members of a multicast group will be senders. We expect that the limited number of active pairs and the use of host clustering due to hierarchical routing will allow HMTP to have a better delay ratio in practice.

3.4.2.4 Link Load

Fig. 3.15 compares the link load of HMTP and Unicast Star in a multicast tree connecting a group with 100 members. Though most of the links have a load of 1, Unicast Star has very high load on a few links. The worst case happens at the last hop to the source, where the link load is almost the same as the group size, as expected. If one link is congested because of high link load, all the downstream members see performance degradation. HMTP avoids this problem by reducing worst case link load significantly. Under HMTP, an end host can control the load on its last hop link by adjusting the number of neighbors it connects to. We

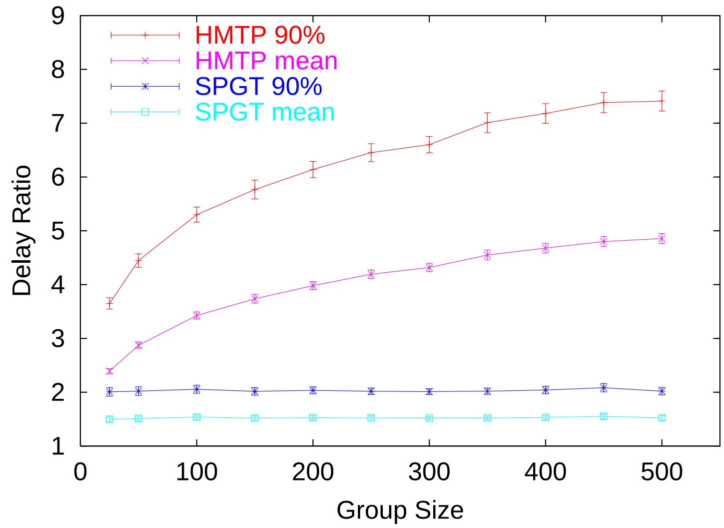


Figure 3.11: Tree delay of HMTP and SPGT tree vs. Group size, with last hop delay 3ms.

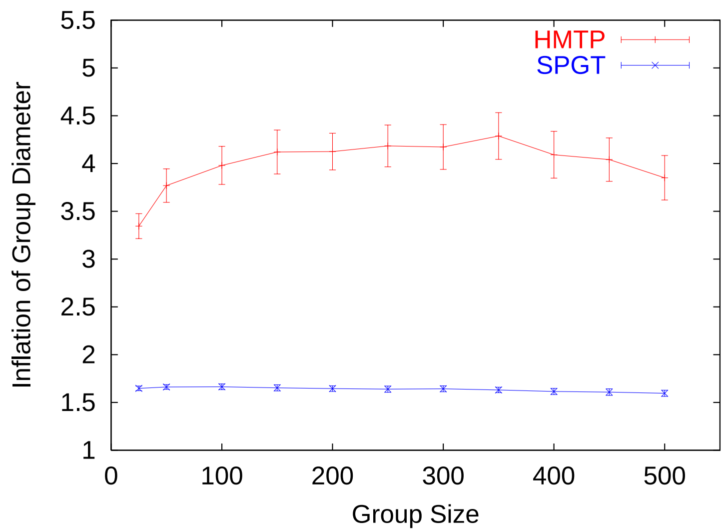


Figure 3.12: Inflation of group diameter for HMTP and SPGT tree. Last hop delay is 3ms.

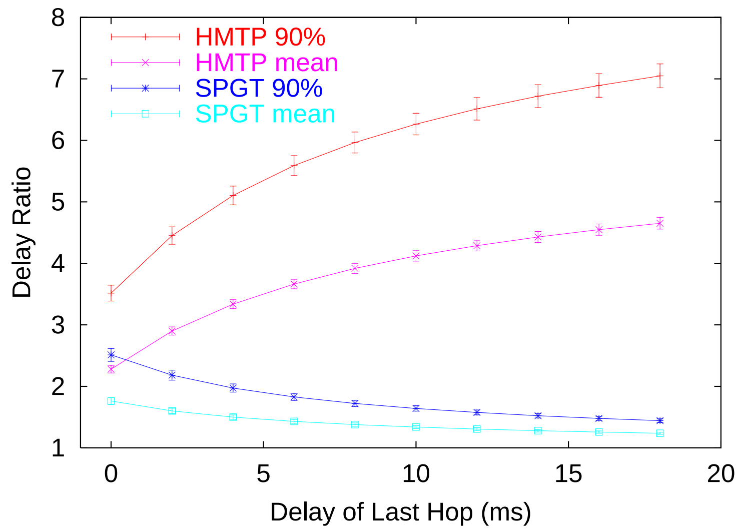


Figure 3.13: Tree delay of HMTP and SPGT tree vs. Last hop delay, with 100 members.

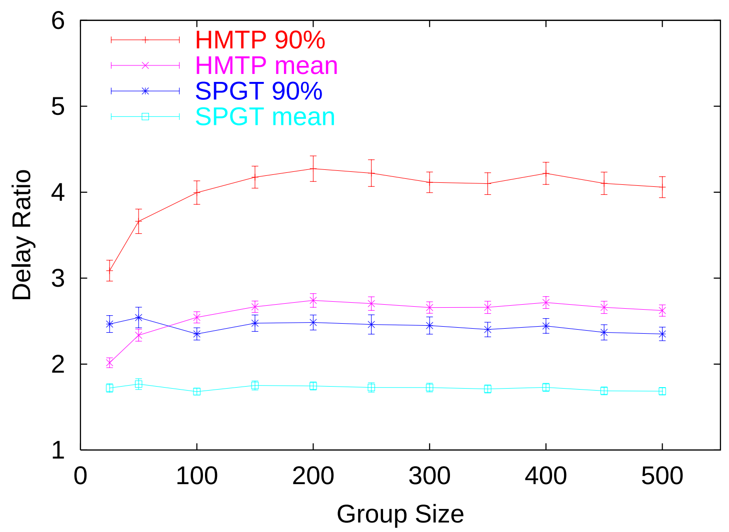


Figure 3.14: Network Delay Ratio vs. Group size. Last hop delay is 0.

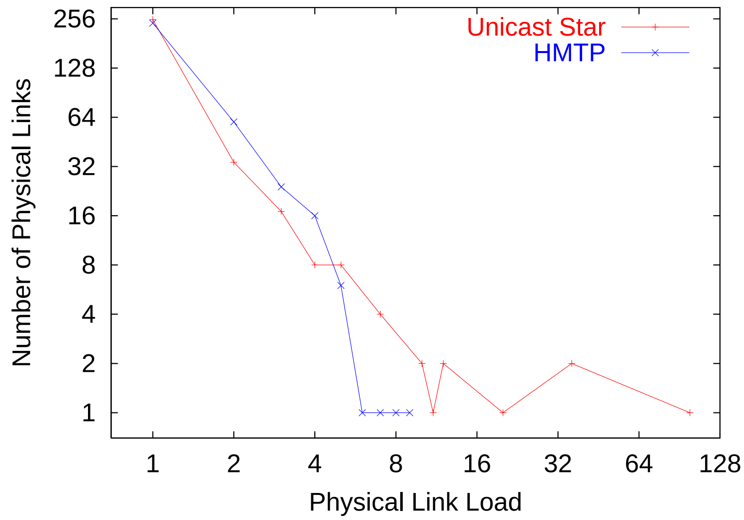


Figure 3.15: Link Load of a HMTP tree and Unicast Star, with 100 members.

use a maximum degree constraint of 8 throughout our simulations. In reality, the degree constraint can be adjusted according to available bandwidth. Smaller degree constraint results in a deeper tree, which may have a higher cost and higher delay, but with lower load on physical links.

3.4.2.5 Experiment

In addition to simulations on random topologies, we also run simulations on a snapshot of real Internet topology. To construct the Internet topology, we downloaded one day's (March 1st, 2001) worth of traceroute data from the NLANR site [Nat]. The traceroutes were conducted among every pair of more than 100 US sites. After removing incomplete measurements, we obtained a topology consisting of 978 routers and 96 hosts. We then ran 1000 simulations on this topology using the same parameters as the simulations we ran on the randomly generated topologies. Comparing the trees constructed by HMTP against that constructed by IP multicast[3] we computed values for our various metrics, averaged over the 1000 runs. Our results are:

- tree cost ratio: 0.99,
- mean delay ratio: 1.76,
- 90%-tile delay ratio: 2.50,
- group diameter inflation: 2.40, and
- worst link load: 8.

[3] IP multicast does not create a minimum spanning tree, which explains the cost ratio below 1 in our result.

CHAPTER 4

Host Group Management Protocol (HGMP)

In HMTP or any other end-host multicast protocol, the basic assumption is that every member host is isolated in terms of network multicast connectivity. Considering IP Multicast is available in every Ethernet, some campus networks, enterprise networks, and a few service networks, the assumption of all hosts are isolated is certainly too pessimistic in the current Internet, not to mention in the future. Even though network multicast does exist in reality at various scales, in order to keep full deployability, end-host multicast protocols cannot take advantage of it to improve performance of data delivery.

When network multicast connectivity exists among certain member hosts, a better trade-off would be using native multicast for local communication and unicast tunnels to reach remote hosts. For example, if there are several members residing in the same Ethernet, it is more efficient for them to communicate via native multicast and select only one member to build unicast tunnels, rather than every host builds tunnels and receives unicast packets ignoring network multicast. Therefore, there are two types of members in a multicast island: Designated Members (DM) who run an end-host multicast protocol for inter-island routing and packet forwarding, and normal members who just send and receive via native multicast. This calls for an intra-island management protocol, Host Group Management Protocol (HGMP), to elect DMs from normal members in every island and coordinate DMs if there are multiple per island.

HGMP expands every single node in an end-host multicast into a multicast island. On one hand, HGMP must retain the deployability. It cannot require any administrative configuration in DM election, DM coordination and island management, which would be much easier otherwise. On the other hand, HGMP must also allow the evolution of multicast service, being able to automatically take advantage of any support, such as network multicast, dedicated server etc., if available.

One important feature we achieved in the design of HGMP is its *protocol independence*: HGMP assumes the existence of an intra-island multicast protocol (e.g., PIM) and an inter-island multicast protocol (e.g., HMTP), but it does not make any assumption on which particular protocol is in use as long as they provide multicast functionality at network layer and application layer respectively. In practice, network operators choose intra-island multicast protocol, and applications choose inter-island multicast protocol. Since both network multicast and end-host multicast are in constant development and deployment, protocol independence gives HGMP great flexibility to accommodate different network environments and future progresses.

In this chapter, we first review some common techniques used in HGMP design, then discuss the scenario that there is only one DM per island, followed by the discussion of multiple DMs per island. We also discuss how to react to network partition and merge. Finally we use simulation to show the performance improvement due to HGMP.

4.1 Background On Multicast Techniques

As an intra-island management protocol, HGMP relies on the availability of IP Multicast within the local island. Besides regular multicast transmission, HGMP also uses some techniques, such as leader election, multicast scoping etc., which are common in many multicast protocols and applications. Before getting into HGMP design, we first review these techniques briefly.

Transmission Scoping There are three ways to limit the transmission scope of an IP Multicast packet. The simplest way is for the sender to set a particular TTL value in the IP header of the packet. The TTL value is decremented by one at every router hop, and it will cause the packet to be dropped when it reaches zero. The area that the packet can reach is a circle (in terms of router hop) with radius of the initial TTL value. If the sender wants the packet to stay in a particular network area such as a campus network, administrative configuration is required. The original practice is to configure multicast border routers with a TTL threshold. Any packet with TTL less than the threshold will be dropped at the border. For example, if a campus network uses TTL = 16 as the threshold on its multicast border routers, a sender can set its packets with initial TTL = 15 to make sure all packets will remain within the campus network. This TTL threshold scheme does not allow overlapped scope and poses difficulties to dynamic address allocation. A later scheme, called administrative scoping [Mey98], configures multicast border routers with a range of multicast addresses. Packets of a group whose address falls into this range will be dropped at border routers no matter of the TTL value. Administrative scoping is better than TTL threshold scoping in confining group traffic to a pre-defined network area, but it is less deployed in reality, partly due to its management overhead. The IETF MALLOC working group has developed a protocol called Multicast Zone Announcement Protocol (MZAP) [HTK00], which conveys the scope configuration information to hosts and routers, letting applications know which scopes are available and facilitating troubleshooting. HGMP requires scoped transmission, but it does not need perfectly configured scopes. Administrative configuration can help hosts choose a better scope and confine traffic better, but HGMP can still work even with the basic IP TTL scoping.

Leader Election and Service Discovery IP Multicast makes it easy to dynamically elect a leader from a group of participants in a distributed and scalable way. In the simplest form, all participants join a multicast group and set a random timer. When one's timer expires and it has not received any message from others, it will send out its own message. The first one sending out a message wins the election and others will cancel their own message sending. Usually the leader continues sending the message periodically to assert itself in the group. When the leader leaves the group, it sends a quit message which will trigger a new round of election among the rest of the participants. If the leader crashes or the network is partitioned, some or all participants will not receive the message any more and will start a new election too. Besides picking the one who sends out a message first, other election criterion, e.g., smallest IP address, shortest distance to a server etc., can be used too. This leader election technique is scalable in that most participants do not need to send a message and they only need to make local decision based on the election criterion. Another advantage is that the participants do not need to know each other beforehand. The performance (e.g., delay before the first message, number of messages generated etc.) depends on group size, timer distribution and network delay, and is well studied in [Sch00].

A similar technique to leader election is service discovery. A client wishing to discover a server sends the query to a multicast group of which servers are members. A server multicast the reply after a random delay so that both the client and other servers will receive it. The first reply suppresses other server from responding. In the case that the client wants to find a nearby server, it can send the query with small scope first, and increase the scope in subsequent queries until a server is discovered. This technique is often referred as *expanding ring search*.

Group Directory Announce-listen is another technique used in various protocols and applications, such as SAP [HPW00] and sdr [Han98]. In sdr, hosts periodically multicast information about their sessions, e.g., a multimedia conference, to a well-known group. Receivers just passively listen to the group to collect session information from all hosts over time. This soft-state approach eliminates the complexity of dealing with various network faults, since they will be fixed by later arrived announcements. In HGMP, we use the same technique to disseminate all group information in local island and call it group directory. A host can look up a UM group in the group directory to find out the multicast addresses for DATA_GROUP and other local groups. Announce-listen protocol gains simplicity by sending more messages, which means it has to send announcements in a slow rate in order to avoid congestion. This leads to traditional sdr's slow-start problem: a host has to wait for sometime (around 30 minutes in current MBONE) to collect all sessions currently being announced. We will present a solution in chapter 6 for this problem. The group/session directory also helps address allocation. Since a host knows all addresses that are currently being used, it can choose a new address randomly from an unused address range. Currently there are several multicast address allocation schemes, such as GLOP static allocation [ML01], sdr dynamic allocation, and the MAAA [THE00] architecture

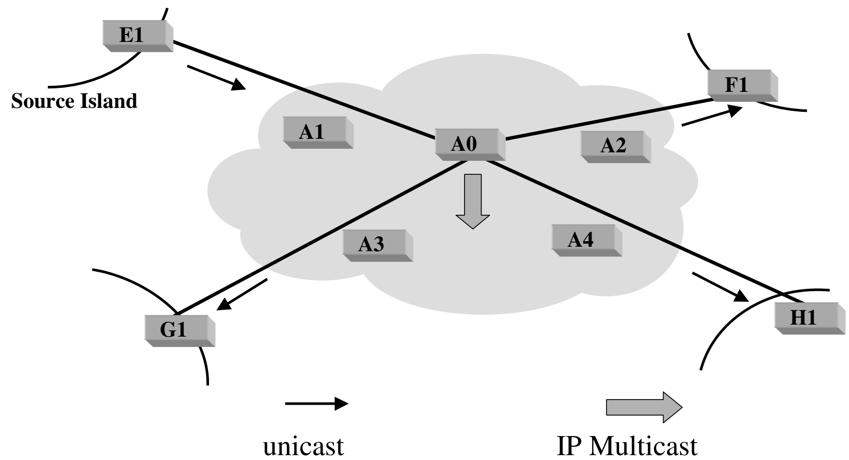


Figure 4.1: Single-DM island

etc. HGMP needs to allocate three local IP Multicast addresses for local groups, but how these addresses are allocated is orthogonal to our design.

4.2 Island with Single Designated Member

When there is only one single DM in an island (Fig. 4.1), it makes no difference to the inter-island routing protocol whether the host running it is an isolated host or is a DM of a multicast island. The only addition is that the DM should do the translation between tunneled unicast traffic and local multicast traffic. Therefore the real question is how group members residing in the same island recognize each other and elect a DM among themselves.

4.2.1 End-Host Only

When a host joins a UM group with the group identifier G, it first checks the well-known group directory in its local island for an announcement for G. If no such announcement is present, it assumes that itself is the first group member in this island and becomes the local DM for G. It then creates three new IP Multicast groups, DATA_GROUP, ASSERTION_GROUP and DM_GROUP, in the local island, associates them with G, and announces the mappings to the group directory (Fig. 4.2). The DATA_GROUP, ASSERTION_GROUP and DM_GROUP are not well-known groups like the group directory. Instead, they are dynamically allocated and specific to each UM group G.

The DATA_GROUP is used for transmitting data packets. Applications send to DATA_GROUP its data packets, which will be received by the DM's agent. The DM then sends these packets out to other islands through tunnels and multicast incoming tunneled packets to DATA_GROUP. Therefore applications still send and receive native IP Multicast packets without bothering how they are transmitted in the network.

The ASSERTION_GROUP is used for DM election. The DM of a group G periodically sends ASSERTION messages to G's ASSERTION_GROUP. All members of G must continuously listen to its ASSERTION_GROUP. When the DM leaves G, it sends a QUIT message to the ASSERTION_GROUP. Remaining members will start an election upon receiving the QUIT message by scheduling sending their own ASSERTION messages after random delays. The first member to send out its ASSERTION message becomes the new DM, and others cancel their message sending. A tie can be resolved by picking the member with the smallest IP address. When the ASSERTION message is continuously missing for a number of periods, a new election will be triggered too.

Since the DM is a normal user's computer, usually there is no incentive to sharing its bandwidth and computing power with others when the owner no longer has interest in the multicast group. Therefore when all applications on local host left the group, the DM will leave the group too. For some types of groups the

DM is likely to change as members join and leave the group continuously. To reduce packet loss during change of DM, the old DM should continue to forward packets after sending its QUIT message for a short period of time. To ensure smooth transition, each ASSERTION message carries information on inter-island routing (e.g., the parent and children nodes on the HMTP tree). With this information, the new DM can quickly establish necessary tunnels to other islands or repair the inter-island multicast tree if the old DM crashes. ASSERTION messages are sent out at a much faster rate than the advertisement rate in group directory to speed up the detection of crashed DM. This is why we use a separate ASSERTION_GROUP for each multicast group G instead of simply relying on the cessation of advertisements in group directory to detect crashed DM.

One enhancement to the basic DM election criterion is to favor hosts that have more resources. Each host has a *priority* computed as a function of its resources. This priority is included in the ASSERTION message. A message with higher priority always wins over, or suppresses, a message with lower priority, regardless of the messages' relative sent order. Therefore a host with Ethernet connection can take over the role of DM from a host with only dial-up connection. However too many priority levels could slow down the election process and also lead to many DM changes as members join and leave the multicast group. Hence we stipulate the use of only a small number of priority levels, e.g., based only on the type of network access technology a host has.

4.2.2 Dedicated Server

Using normal hosts as DM is necessary to meet UM's deployability requirement. Nevertheless, when a dedicated server with more computing power and network bandwidth is available, using it as the DM can improve the performance and

```
Agent:
check the group directory;
if (G is NOT in the group directory) {
    // become the DM
    create DATA_GROUP, ASSERTION_GROUP and DM_GROUP;
    announce G in the group directory;
    send assertion messages to ASSERTION_GROUP
    run inter-island routing protocol to build tunnels to other islands;
}
else {
    // just a normal member
    join ASSERTION_GROUP;
}

Application:
join DATA_GROUP;
```

Figure 4.2: Join Group G (single-DM model)

stability of data delivery. Servers are usually also more stable and properly managed. While a DM of normal host would leave a group when applications running on it are no longer interested in the group, a DM of dedicated server can keep forwarding a group's traffic until the last group member in the local area network leaves the group. This reduces the number of DM changes and enhances the stability of the forwarding service.

A server is configured with an election priority higher than normal hosts' and a service area which it is residing in and willing to serve (e.g., a department network). The server periodically sends scoped multicast messages to its service area to query local membership, and normal members send back reports via multicast too, much like IGMP query/report mechanism. In fact, if the service area is a local area network (LAN) and has a multicast router in it, the server can just passively monitor the IGMP traffic to learn group membership of hosts. When there are members of group G in its service area, the server will participate the DM election by sending its higher-priority ASSERTION messages, so that it will always supersede normal hosts in becoming a DM. A server ceases its role as DM for group G when there is no longer any host in its service area is interested in the group, even if there are still some group members in other network areas. This design decision removes a possible disincentive network operators may have in hosting a UM server. A server DM follows the same departure protocol as normal DM to minimize service disruption.

While a server's assumption of a DM role is based on local membership in its service area, the ASSERTION messages are sent to the entire island. If there are multiple servers each in different service area, they will compete and the one with highest priority will be elected. Backup servers can be configured with a priority higher than normal hosts' but lower than that of the primary server of the same service area. During DM election, backup servers automatically take over if the primary server does not send out an ASSERTION message. If all servers are down, a normal host will be elected as DM according to the election algorithm. Thus the deployment of servers is totally transparent to hosts, except performance improvement experienced by applications.

4.3 Island with Multiple Designated Members

The single-DM model works fine for small IP Multicast islands. For a large IP Multicast island with many group members, it may result in performance problems such as long latency and high traffic concentration. In this section, we extend the simple DM election protocol in previous section to support multiple DM per island, and discuss how the multi-DM model can alleviate performance problems with better utilization of network multicast, and how multiple DMs in an island should coordinate to maintain inter-island routing.

4.3.1 The Need for Multiple DMs

When there is only one DM in an island, all the group traffic coming into or going out of the island must go through the DM. This leads to two potential problems in large IP Multicast islands. For example, in Fig. 4.1, there are five islands each with a single DM. Assume the inter-island routing protocol builds a shared-tree among these islands. The ideal way to send a packet from island E to island A is to let E1 send it to A1, and A1 multicast it into island A. However, in Fig. 4.1, packets from E1 must detour through A0 to enter island A. This results in longer delivery delay for other members in island A. If A1 is chosen as the DM, longer delay will happen to packets coming from island F. The magnitude of the

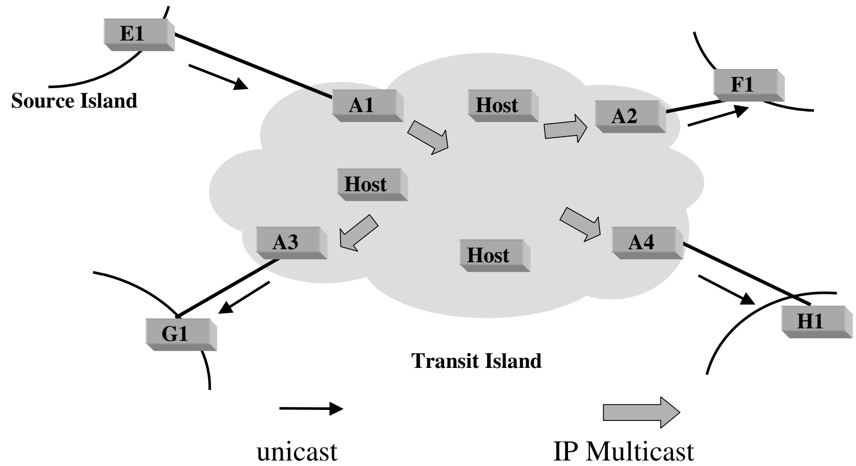


Figure 4.3: Multi-DM Island

delay penalty depends on the size of the island and the distribution of member hosts in the island. When there are many group members scattered in a large island, with multiple inter-island tunnels to other islands in different directions, the DM's location will have significant impact on the performance (e.g., latency) of data delivery. It is hard for a single DM to deliver good performance to all members in such situation.

Another problem is that the single DM is a point of traffic concentration because it has to handle all the traffic coming in and out of the island. In a large island, other members may likely reduce traffic concentration by using alternate paths. For example, in Fig. 4.1, A1 can take care of the tunnel to E1, so that A0 only needs to handle other three tunnels. Under the single-DM model, A1 and A0 cannot be DM at the same time. Both longer delay and higher traffic concentration still exist if the inter-island topology is a per-source tree. Thus these problems are inherent to the single-DM model regardless of the inter-island routing protocol.

A natural solution is to allow multiple DMs per island. In Fig. 4.3, island A has four DMs. A1, A2, A3 and A4 share the workload of packet forwarding as each maintains one unicast tunnel. Packets coming into or going out of island A now take shorter paths than always having to detour through a single DM. The multi-DM model thus solves the problems with the single-DM model. However, it has problems of its own, namely how to elect multiple DMs per island and how to coordinate inter-island routing in the presence of multiple DMs.

4.3.2 Multi-DM Election

In the single-DM model, an ASSERTION message travels the full extent of an IP Multicast island. Thus the DM's ASSERTION messages can effectively suppress all the other members from becoming a DM. To allow multiple DMs per island, we modify the election protocol in two aspects: (1) the assertion scope can be smaller than the data scope, and (2) overlap of assertion scopes must be tolerated.

When the scope of ASSERTION messages is smaller than the scope of data packets, members outside assertion scope will not hear the ASSERTION message, hence they will start electing a new DM among themselves. By this way, multiple DMs can be elected automatically, each with an assertion coverage around its network neighborhood. These DMs, however, share the same island-wide data scope, which means a data packet multicast by one DM will reach all group members.

The ideal way to set assertion scope is to let every DM dynamically adjust its assertion scope, so that there is no overlap between any two assertion scopes while the entire island is covered by the union of all assertion scopes. However, it is hard to achieve this goal by using IP Multicast scoping. When an overlap happens and one DM wishes to give up the overlapped area, it is almost impossible to specify a new scope which is "the same as the old one except for the overlapped

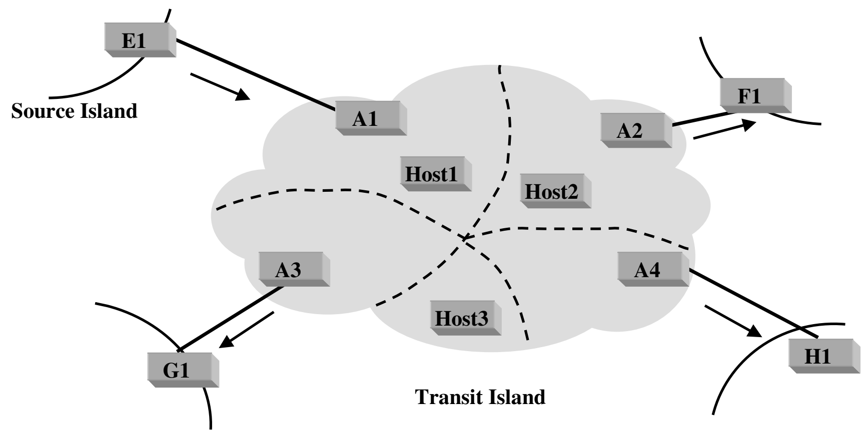


Figure 4.4: Election of Multiple DMs

area." Because IP Multicast scope is either a circular coverage (i.e., generic IP TTL scoping) or administratively configured (TTL threshold and administrative scope), a host cannot find a scope to cover a network area arbitrarily shaped. Therefore we decide to tolerate overlap of assertion scopes. The only overhead of this is redundant assertion messages in the overlapped areas, which should be low volume traffic. There will be no duplication of data packets since HGMP ensures that data packets enter an island only once and only via a single DM (Section 4.3.3). Not requiring perfectly separated scopes enables HGMP to use any multicast scoping scheme available, which is in line with our deployment requirement. Better scoping schemes like administrative scope gives HGMP more choices in setting a suitable assertion scope.

How to choose an assertion scope appropriate to a given network environment remains an open issue. Very small assertion scopes may generate excessive number of DMs, which will complicate the inter-island topology and cause overhead to routing. Appropriate assertion scopes should give rise to a number of DMs adequate to utilize the large island, yet not excessive to incur much overhead in inter-island routing. In an ASSERTION message, its initial IP TTL value is carried in the payload. By comparing the initial TTL value and the received TTL value, a receiver can estimate how many hops away a DM is. If a DM sees another DM which is very close by, it can cease to be a DM. Conversely, if a normal member finds that its current DM is too far away, it can become a new DM. In this way, even if a DM is mis-configured with an assertion scope too small or too large, others can offset the effect to maintain an appropriate number of DMs in an island. In current implementation, DMs use a pre-configured assertion scope corresponding to organization scope. More experiments and practical experiences are needed to refine how to set the assertion scope.

4.3.3 Inter-Island Routing with Multiple DMs

4.3.3.1 The Problem

DM election is just one part of the design for multi-DM model. The other part is how these multiple DMs coordinate with each other to ensure correct inter-island routing. A single-DM island has only one point for packets entering or exiting the island, therefore an island can be abstracted as a single node in inter-island topology. In contrast, a multi-DM island can have multiple entrances and exits for packets. Without proper coordination among the multiple DMs, an island cannot be abstracted as a single node, which breaks the assumption of all end-host multicast protocols and causes problems to inter-island routing. If we let each individual DM operate independently as before, the multicast service will collapse due to routing loops and packet duplicates. Fig. 4.5 illustrates the problem as follows.

Assume island A's DMs, A1 through A4, run an end-host multicast protocol independently and build tunnels to other islands as shown in the figure. When

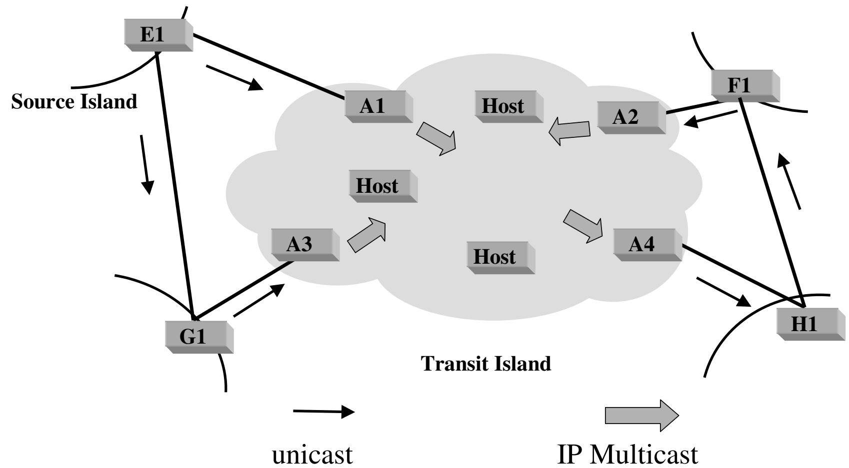


Figure 4.5: Routing problems for multi-DM island

a data packet is originated in island E, E1 sends it to both A1 and G1, and the latter will forward the packet to A3. Not aware of the other's behavior, A1 and A3 will both accept the same packet and multicast it into island A causing duplicates. When the multicast packet reaches A4, it will be forwarded to H1, F1 and back to A2, resulting in more duplicates in island A due to the routing loop. The packet forwarding also loops along the other direction (A2-F1-H1-A4) and another path (i.e., both directions of A1-E1-G1-A3). The coupling of packet duplicates and routing loops produces large volume of redundant traffic whose only bound on lifetime is the packet's TTL value and will jam the network and bring down the service eventually.

In an multi-DM island, IP Multicast becomes a transit part of the data delivery path, rather than just the last hop as in a single-DM island. The multiple DMs in the same island are connected by IP Multicast implicitly, but they are not aware of each other. This *invisible* connectivity breaks the inter-island routing regardless of which end-host multicast protocol is used.

Our solution to the problems resorts to explicit coordination among multiple

DMs. HGMP needs to interact with the inter-island routing protocol in order to prevent routing loops and packet duplicates. One challenge is how to interact with the routing protocol and be protocol-independent at the same time. We categorize end-host multicast protocols into two general types according to the multicast distribution tree in use: shared-tree protocols and source-tree protocols. Our solution depends only on which type of protocol is used, not any technical details of the particular protocol. We believe this categorization is general enough to include all end-host multicast protocols and specific enough for the coordination and interaction to be effective.

For any solution to this multi-DM inter-island routing problem, we wish it can fully utilize the multiple DMs in addition to eliminating loops and duplicates. That is, in an ideal solution, a packet originated inside the island exits through multiple DMs, and a packet coming from outside enters the island through multiple DMs too. The latter is practically difficult since it requires DMs multicast the packet with transmission scopes that cover all member hosts as a whole and without overlap between each other to avoid duplicate data packets. As discussed in 4.3.2, IP Multicast scoping is not flexible enough to implement this at all the time. Therefore, the goal for our solution is to allow multiple exits and single entrance for each packet with no loop nor duplicate. Though one packet enters an island via only one DM, different packets may use different DMs, which still benefits from the presence of multiple DMs.

4.3.3.2 Head DM and Island Number

The communication among multiple DMs is done via the third local IP Multicast group, DM_GROUP. The DM_GROUP has all DMs in the same island as its members and has the same scope as the DATA_GROUP. DMs use this multicast group to dynamically elect one DM as the *Head DM*, while others are *Tail DM*. The Head DM periodically sends ALIVE messages to DM_GROUP and sends a LEAVE message before it leaves the group. Tail DMs start a round of election upon receiving the LEAVE message or missing ALIVE message continuously. In source-tree protocols the DM who sends ALIVE message first becomes the new Head DM, while in shared-tree protocols the election criterion takes into account of network delay (Section 4.3.3.3). The Head DM has several functions. First, it is responsible for announcing this UM group periodically to the group directory. Second, the Head DM is needed for maintaining correct inter-island routing in shared-tree protocols (Section 4.3.3.3). Third, its ALIVE message serves as a signal of network connectivity, so that Tail DMs can detect network partitions and take corresponding actions (Section 4.4).

Many end-host multicast protocols, especially source-tree protocols, need to uniquely identify each island in the routing table, usually by using the DM's IP address (Section 4.3.3.4). In a multi-DM island, all DMs should agree on which identifier to use. Under IP Multicast, an island's address prefix can be used as its globally unique identifier. However, under end-host multicast, a host usually does not know how large its island is, nor how many address prefixes its island encompasses. Our solution is to assign an *island number* to each island. The simplest way to obtain an island number is using the Head DM's IP address. Other ways include using Head DM's FQDN (Fully Qualified Domain Name), or having UMRP assign a number when the Head DM contacts it at start time. As long as the island number is unique for each island in a UM group, applications and inter-island routing protocols can have their own ways of assignment. Once the island number is available, it is included in the Head DM's ALIVE messages so that Tail DMs will know it. Besides being used in source-tree inter-island routing, island number is also used in dealing with network partitions and merges (Section

4.4).

4.3.3.3 Shared-Tree Protocols

In protocols like BTP [HJ02], HMTP [ZJZ02] and Yoid [Fra01], all participating nodes are self-organized into a single tree structure, which is shared by all data sources. Multicast delivery is achieved by simply flooding this shared-tree: an incoming packet is forwarded on all links except where the packet comes from. Though different protocols have different ways of constructing the shared-tree, here we are only interested in how to keep such a protocol working when each node is expanded to a multi-DM island, regardless of any protocol-specific detail.

The reason that flooding the shared-tree does not cause any routing loop or packet duplicate is because of the tree structure. With multiple DMs in the same island, intra-island IP Multicast connects these multiple DMs in a way equivalent to full-mesh connectivity, which violates the tree structure and breaks data forwarding. The solution is to organize an island's multiple DMs into a two-level "mini-tree". The Head DM runs the inter-island routing protocol as usual to find its parent in other islands. Tail DMs, on the other hand, must always take the Head DM as their parent, as illustrated by the dotted lines in Fig. 4.6 between Head DM A1 and Tail DMs A2, A3 and A4. From another island's point of view, island A now has only one parent (i.e., E1) and multiple children (i.e., F1, G1, and H1), just as a single-DM island. When all DMs run the inter-island routing protocol with this restriction, there will be no link E1-G1 or F1-H1 in Fig. 4.6, because they form loops explicitly considering the dotted lines, and the routing protocol will not allow them. Therefore, the tree structure, composed of both unicast tunnels and IP Multicast, is preserved through multi-DM islands, which ensures that the same forwarding scheme can be used without routing loop or

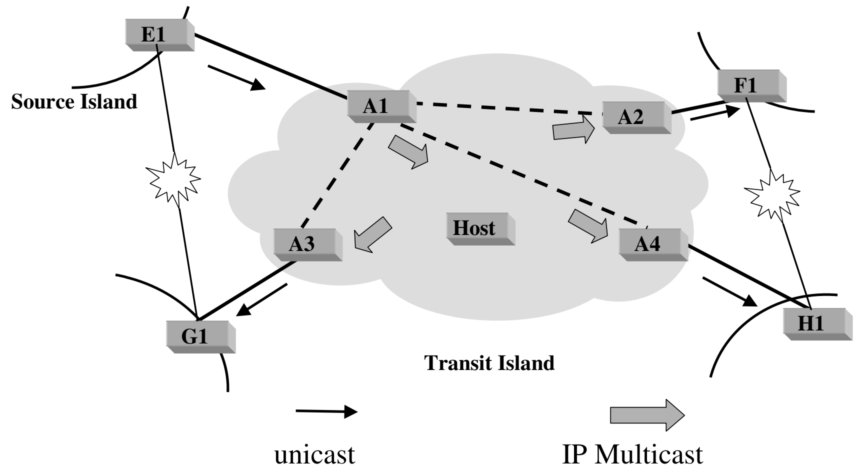


Figure 4.6: Multi-DM island with shared-tree inter-island routing

packet duplicate. Note that the parent-children relationship between Head DM and Tail DMs is only logical. There is no actual unicast tunnels between them as data packets are still transmitted via IP Multicast inside the island.

Initially, the first DM in an island becomes the Head DM (Fig. 4.7). When there are multiple DMs, the election of Head DM takes into account of network delay. Every DM runs the inter-island routing protocol independently to decide its potential parent in other islands and measures RTT (Round-Trip Time) to the potential parent. The Head DM includes its RTT value in its ALIVE messages. If a Tail DM has a smaller RTT than that advertised by the Head DM, it assumes the role of Head DM by sending its own ALIVE message with the smaller RTT value. The result is that the Head DM is always the DM with the shortest RTT to its extra-island parent. This election criterion favors DMs at island edge over DMs in island center, thus IP Multicast is utilized better as packets are multicast at the earliest point into an island. For this purpose, each island only needs to minimize the delay of Head DM's extra-island link, since a Tail DM's extra-island neighbor must be a Head DM of another island.

```
Agent:
check the group directory;
if (G is NOT in the group directory) {
    // become Head DM
    obtain island number;
    create DATA_GROUP, DM_GROUP and ASSERTION_GROUP;
    announce G in the group directory;
    send ASSERTION message;
    send ALIVE message;
    run inter-island routing protocol;
}
else {
    join ASSERTION_GROUP;
    if (no assertion message is heard) {
        // become a Tail DM
        send ASSERTION message;
        join DM_GROUP;
        take Head DM as parent;
        run inter-island routing protocol;
    }
    // else just a normal member
}

Application:
Join DATA_GROUP;
```

Figure 4.7: Join Group G (shared-tree routing)

4.3.3.4 Source-Tree Protocols

Source-tree protocols, such as CAN [RHK01], Narada [CRZ00], SALM [BBK02] and TMesh [WHJ02], build inter-island topologies more richly connected than a single tree, thus data from different sources may use different distribution trees. Since there is no shared-tree structure, source-tree protocols need packet forwarding schemes more sophisticated than simple flooding. Though existing protocols look very different from each other, there are two elements that they all have in common: (1) Each node exchanges some routing information with its neighbors to build own routing table; and (2) a local forwarding rule decides where to forward a packet based on the routing table and where the packet comes from. In Narada and TMesh, the routing information is shortest path vector and the forwarding rule is Reverse Path Forwarding (RPF)[1]. CAN and SALM use node's logical coordinates (e.g., dimension number or layer number) as routing information and have very different forwarding rules. Regardless of the actual contents of routing information and forwarding rules, the invisible multicast connectivity in a multi-DM island makes the routing table incomplete, breaking any forwarding rule used by the protocol. The solution is using island number to identify each island in the routing table and using DM_GROUP to exchange routing information within an island to complete the routing table.

A source-tree protocol needs to identify each island in its routing table and routing exchange. In a multi-DM island, DMs should all agree on the same identifier. As discussed in Section 4.3.3.2, we use island number as such an identifier, which can be the Head DM's IP address or assigned by application-

[1]In RPF, a packet originated by source S is forwarded by F to a receiver R only if R uses F as the next hop in R's shortest path to S. The distribution tree generated by RPF is the reverse shortest path tree rooted at the source. This technique is used in some IP Multicast routing protocols like DVMRP and PIM-DM.

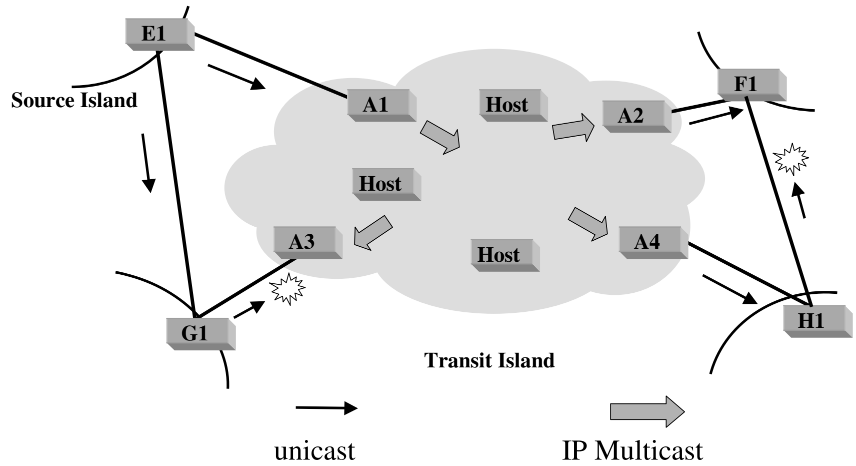


Figure 4.8: Multi-DM island with source-tree inter-island routing

specific ways.

In additional to regular routing exchange with neighbors in other islands, DMs also periodically multicast their routing exchange to the DM_GROUP so that all DMs will get complete routing information. Take Fig. 4.8 as an example and assume Narada is used as the inter-island routing protocol. After the inter-island routing exchange and intra-island routing exchange, all DMs in island A (i.e., A1, A2, A3 and A4) know that the shortest path from island E to island A is via link E1-A1. Similarly, F1 knows the shortest path from island E goes through A2, and H1 knows the shortest path from island E goes through A4. When E1 sends a packet, the copy going through E1-G1-A3 will be dropped because it fails the RPF checking. Similarly, the copy between H1 and F1 will be dropped too. Now, for the purposes of inter-island routing and packet forwarding, every island can be viewed as a single node represented by its island number, and every DM has complete routing information to apply the forwarding rule correctly.

This problem and solution of source-tree protocols in multi-DM islands bear similarity to inter-domain routing on the Internet using BGP. One difference is

Agent:

```
check the group directory;
if (G is NOT in the group directory) {
    // become the first DM
    create DATA_GROUP, DM_GROUP and ASSERTION_GROUP;
    get island number;
    announce G in the group directory;
    send assertion message;
    run inter-island routing protocol;
    advertise routing table;
}
else {
    join ASSERTION_GROUP;
    if (no assertion message is heard) {
        // become a DM
        send assertion message;
        join DM_GROUP;
        run inter-island routing protocol;
        advertise routing table;
    }
    // else just a normal member
}
```

Application:

```
Join DATA_GROUP;
```

Figure 4.9: Join Group G (source-tree routing)

that the UM island number is dynamically assigned as group members come and go, compared to the AS number which is administratively managed. Another difference is that HGMP uses IP Multicast for intra-island routing exchange, while iBGP (internal BGP) uses unicast.

4.3.3.5 Thoughts On Alternatives

There are some conventional techniques that can identify a loop by looking at the packet only. This is appealing because it does not need a new protocol among forwarding nodes and is easy to implement comparing to our approach of explicit coordination among DMs. Unfortunately they do not work well for the multi-DM islands as described below. Furthermore, its overhead increases as the number of data packets increases, leading to possible performance degradation when traffic volume is high. On the other hand, the overhead of our approach is mainly due to periodic routing announcements and updates, which is almost constant regardless of how much data traffic is in the network.

One technique commonly used to prevent routing loops and suppress packet duplicates is for the original source to tag every packet with a monotonically increasing sequence number. A forwarding node remembers recently received packets and drops any incoming packet if it has seen it before. However, this technique cannot be directly applied in our case of multi-DM island. When a packet is originated by a host inside an island, it is multicast by the application and we cannot assume or enforce the application to put a sequence number on it. It should be the DMs who put a sequence number in the encapsulation header when they forward the packet out via tunnels. But multiple DMs are in different locations, have different network connections, different packet loss patterns and may receive packets in different orders. These make it impossible for multiple DMs to come up with the same sequence number for the same packet without proper coordination. Therefore, for an island with n DMs, one packet can potentially generate n duplicate copies with each having a different sequence number.

Another way to identify a packet is looking at the ID field of the IP header, or hashing the IP header and the first few bytes of the payload. Multiple hashing functions may be used as in Bloom Filter [?]. Even if this method works, it can only eliminate routing loops not packet duplicates. For example, in Fig. 4.5, when E1 sends out a packet, this method cannot prevent the packet from entering island A twice via A1 and A3 respectively. Besides the hashing calculation, another overhead is that the receiver has to actually receive the packet in order to determine whether to drop it or not, by then the bandwidth between the sender and receiver has already been consumed.

Recording the path that a packet has traversed in the packet header is another common technique to prevent loops. Every forwarding node puts its own IP address in the packet it forwards, so that the receiver can determine whether it has seen a packet before or not. In a multi-DM island, using DM's IP address is not effective since the packet can still loop back to the same island via a different DM. Therefore the island number must be used when recording the path. But like the sequence number approach, the best this method can do is eliminating routing loops. It cannot eliminate packet duplicates.

4.4 Network Partition and Merge

HGMP deals with network faults with a soft-state approach by periodically sending control messages such as ASSERTION messages and ALIVE messages. When

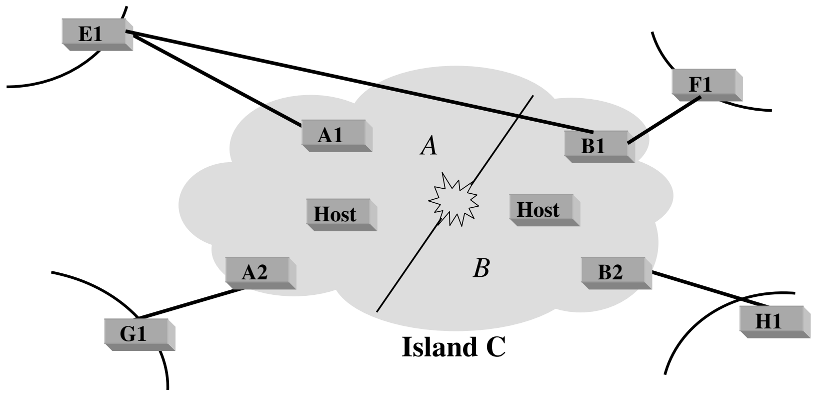


Figure 4.10: Network Partition and Merge

a fault happens, either a later arrived control message repairs the fault (e.g., packet loss), or the receivers time out and start repairing the fault (e.g., network partition, host crash) by themselves. Before the fault is repaired, multicast delivery service can be negatively affected during the transition period. In this subsection we study how to minimize the damage of network partition and network merge during the transition period.

Network partition may split one island into two islands, and network merge may join two islands into one island. Network partition and merge require changes of end-to-end connectivity, which usually involves multiple physical connections. Therefore they are relatively rare in reality since IP routing has built-in mechanisms to deal with link failures and link additions. Nevertheless, when they do happen, inter-island routing is temporarily in an inconsistent state, in which both solutions in 4.3.3.3 and 4.3.3.4 are ineffective, and routing loops and packet duplicates are possible.

In HGMP, the Head DM periodically sends ALIVE messages to DM_GROUP. The continuing loss of ALIVE messages indicates a network partition or Head DM crash, and ALIVE messages from different Head DMs indicate a network merge. When it is detected, HGMP will adapt to the network change by electing one and only one Head DM in the island as discussed before. Since we cannot tell whether the loss of ALIVE messages is caused by network partition or Head DM crash, HGMP adopts the following rule :

- If the DM election is due to the loss of ALIVE messages, as opposed to the arrival of LEAVE messages, the new Head DM should obtain a new island number.

This makes sure that if the island is indeed partitioned, the new island will have its own island number.

To detect network changes faster, short interval between ALIVE messages must be used. This is possible in HGMP since we use different local IP Multicast groups (DATA_GROUP, ASSERTION_GROUP and DM_GROUP) to separate different group traffic. An alternative design choice would be to use the same IP Multicast group but different ports to demultiplex traffic. However, it means the control traffic including ALIVE messages would be delivered to not only the DMs but also normal members, and the ensuing transmission overhead would prevent control messages from being sent in a fast rate. Another important factor to shortening the transition period is how fast the inter-island routing converges, which is orthogonal to HGMP design.

During the transition period, if there a routing loop exists, there will be a large number of packet duplicates (Section 4.3.3.1), which can cause serious traffic jam if the application is sending fast at the time. Therefore, even the transition period is shortened by adopting frequent ALIVE messages, we still need a way to limit the damage caused by the inconsistent routing state in the transition period. For this purpose, we add the following two rules for DM in packet forwarding:

- When a DM multicast a packet into local island, it puts its own island number in the *Current Island* field of the packet's encapsulation header.
- When a DM or a normal member receives an encapsulated packet via multicast in local island, it drops the packet if its *Current Island* field does not match the receiver's island number.

For example, in Fig. 4.10 , suppose island C already has some group members at the time when it is partitioned to two islands, A and B due to a network outage. The inter-island routing takes a while to converge. Then after some time the two islands merge back as the network problem is fixed. During the period that immediately follows the merge but before the inter-island routing converges again, members in A and members in B have different island numbers, though they are actually in the same island C. Outside members (e.g., E1) also think A and B are two different islands before the inter-island routing converges. Since members in A and B use same local IP Multicast groups allocated before island C was partitioned, there will be a temporary loop E-A-B-E causing a lot of packet duplicates. However, with the aforementioned rules, the number of duplicates will be reduced significantly. When a packet is sent out by E1, it enters island C twice from A1 and B1 respectively. The copy from B1 will be discarded by A1 and other members in A because its current island field does not match A's island number. For the same reason the copy from A1 will be discarded by B1 and other members in B. Thus for a packet from outside, there will be only two copies transmitted in the island network and one copy delivered to the application. For a packet originated from inside island C, it will reach both A1 and B1. Since it is generated by the application, A1 and B1 both assume it is from local island and forward it to E1. E1 will forward one to A1 and the other one to B1, which is equivalent to originating a packet by E1 itself. Thus for a packet originated from inside, it will cause three copies transmitted in the island network and two copies delivered to the application. Note the copy forwarded out by A2 will not be able to enter B because one packet enters an island only once (Section 4.3.3.5). Considering it is inevitable for severe network changes to have negative impact on multicast delivery and the transition period is already shortened, a few packet duplicates should not be a problem for the network and application.

In the case that two merging islands were not connected before (e.g., due to newly deployed IP Multicast), group members in the merged island will probably still use two different sets of local IP Multicast groups since they were allocated independently in the original islands. To consolidate them into a single set of local IP Multicast groups requires the application to change multicast group, i.e., DATA_GROUP, which some applications may not be able to do. HGMP takes no action in this case since the traffic in different local groups do not overlap and inter-island routing is not affected.

4.5 Simulation

We conducted simulations to evaluate the performance gain of the multi-DM model over the single-DM model. We use the Internet AS map [CCG02] to represent the inter-domain topology. In each simulation, several ASs are randomly chosen as multicast islands. Each island is then populated with router network using the Waxman topology generator. Finally, host nodes are randomly attached to routers as multicast group members. The number of routers in an island varies from 50 to 500, but is the same for all islands per simulation. The results presented here are from simulations of 100 islands with 20 group members in each island. For each physical network link, we assign delay uniformly between 3-10ms. We have also run simulations with other settings, obtaining qualitatively

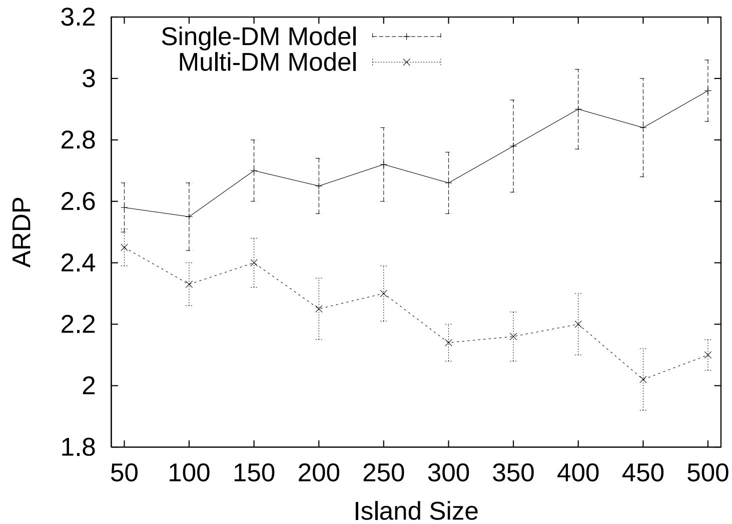


Figure 4.11: ARDP v.s. Island Size (HMTP)

similar results.

Intra-island IP Multicast routing assumed per-source shortest path tree similar to that used in DVMRP. For inter-island routing, we simulated two protocols: HMTP for shared-tree routing and Narada for source-tree routing. In the single-DM model, the DM is randomly selected from member hosts, with uniform probability. In the multi-DM mode, half of the members in an island are randomly selected as DMs, again with uniform probability.[2]

We use two metrics in our performance evaluation: Average Relative Delay Penalty (ARDP) and worst node load. Relative Delay Penalty (RDP) is the ratio of multicast delay to unicast delay between two nodes. ARDP is the average of RDP over all host pairs. The load of a node is the number of unicast tunnels

[2]Clearly we do not expect half of an island's population to serve as DMs. Since in our simulations DM that are not very well placed, performance-wise, will not be selected by the end-host multicast protocol to be on the inter-island multicast tree, their existence does not affect the performance metrics studied. It does mean, however, that the performance numbers reported correspond to cases in which we can find well-placed hosts to serve as DMs. This is a topic of our future work.

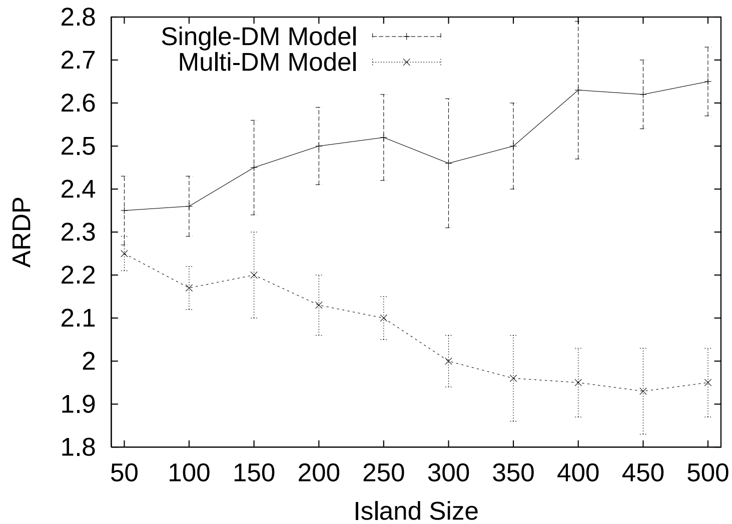


Figure 4.12: ARDP v.s. Island Size (Narada)

it must support. For each simulation, we take the average of 100 runs and plot each data point with 95% confidence interval.

Figs. 4.11 and 4.12 show that as the island size increases, ARDPs under the single-DM case increases whereas those under the multi-DM case actually decreases. We attribute this to the increasing significance of intra-island latency as the island size increases. The multi-DM model can factor out large intra-island delays in inter-island latencies. This is true for both inter-island routing protocols we have simulated. Figs. 4.13 and 4.14 show that while having multiple DMs per island clearly reduces the maximum node load, for the scenarios simulated the effect of island size on maximum node load is not apparent.

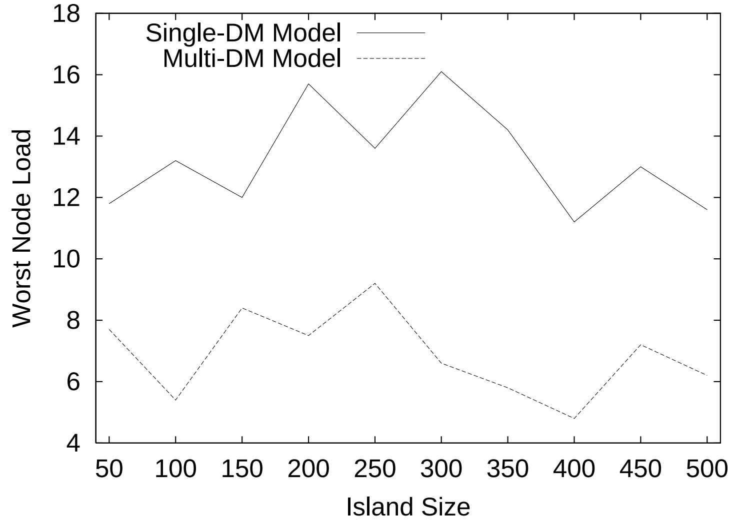


Figure 4.13: Worst Link Load v.s. Island Size (HMTP)

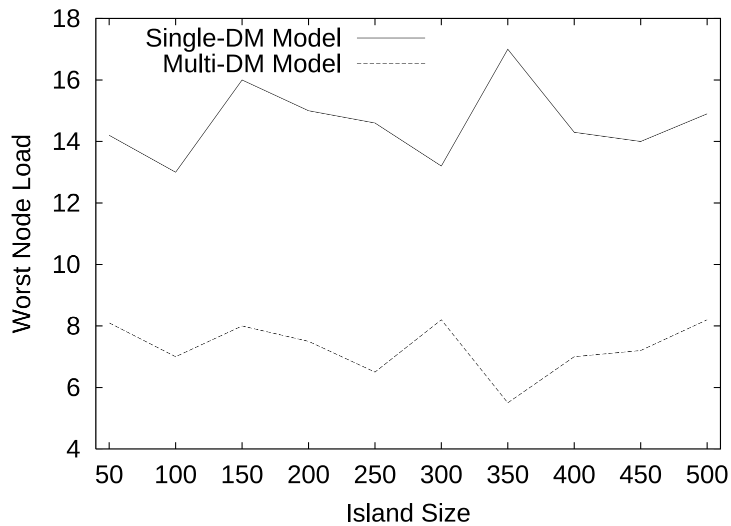


Figure 4.14: Worst Link Load v.s. Island Size (Narada)

CHAPTER 5

UM Agent

The Agent is a daemon program that implements all UM functionalities and runs on every member host. Since UM uses both end hosts and routers in packet delivery, its implementation involves some issues not presented in other multicast schemes.

5.1 Packet Forwarding

Networking functionalities are grouped into different layers forming a protocol stack. A lower layer provides certain services to the upper layer through well-defined APIs without exposing unnecessary details; An upper layer implements its own functionality based on services provided by the lower layer. Layering reduces system's complexity, but sometimes is not flexible for new protocols or applications.

Different from existent protocol stack, in which packet forwarding is done at network layer only, UM forwards packets both by routers at network layer and by end hosts at application layer. The transition between two layers happens at the DM when IP Multicast packets from local island are received and when tunneled packets are multicast into local island. The challenge is for the Agent to preserve some critical information across layer transitions and provide the same delivery service model and semantics as native IP Multicast's, with no changes to operating systems and minimal changes to applications.

5.1.1 Packet Capture

Applications send their data packets to the DATA_GROUP via native IP Multicast. Other members in the same island will receive the packets automatically since their applications are listening to the DATA_GROUP and the application UDP port. In addition, the DM's Agent should be able to receive these packets so that it can forward them to other islands. Two mechanisms have been used in previous works for receiving packets in similar situations, but they are not suitable for UM.

One mechanism is using IP-in-IP tunnel as in MBone. The operating system is updated so that it can recognize incoming multicast packets, append a new unicast IP header, consult forwarding table, and send them to other islands. Since it operates at network layer, which is within operating system kernel, using IP-in-IP tunnel would compromise UM's deployability.

Another mechanism is using UDP sockets as in mTunnel [PSS98] and UMTP [Fin01]. The Agent opens a UDP socket listening to DATA_GROUP and application UDP port to receive packets, just like another instance of the multicast application it is serving. Though easy to implement, it has several drawbacks. First, the Agent can only receive and forward packets destined to a particular port, not all packets destined to the DATA_GROUP. As IP Multicast routers forward all packets of a group regardless of UDP port, the UM Agent should operate in the same way to provide the same delivery service to applications. Second, the application has to make some changes. It needs enable multicast loop-back for the Agent to receive packets from local host, inform the Agent of its UDP port, which sometimes may be dynamic, and set a socket option to allow the Agent's UDP socket to bind the port. Third, a non-member sender's Agent needs to listen to DATA_GROUP for the purpose of receiving packets from applications on local host. But as a side-effect, the host becomes a member of DATA_GROUP and starts receiving group traffic from other sources, which is not needed by any application.

In our implementation, the UM Agent uses packet capture, sometimes called "packet sniffing", to receive multicast packets from local island. Most operating systems allow applications receive copies of raw packets directly from the network interface. Using this mechanism, the Agent is able to receive all packets destined to DATA_GROUP regardless of the UDP port, and avoid the side-effect of joining DATA_GROUP on non-member hosts. Applications do not have to turn on multicast loop-back, inform Agent about UDP port, or set additional socket options. Furthermore, the Agent can obtain some lower-layer's information, such as IP TTL, which is not available through the socket API.

Since the group traffic of the Agent's interest is either delivered to the host if it is a group member, or originated from the host if it is not a member, the Agent does not need to put the network card in promiscuous mode[1] to sniff other packets, relieving it from high workload in most general-purpose packet sniffers, which capture all packets on the wire. The performance of this approach depends on how operating systems support the packet capture mechanism and how network cards filter packets. It can be implemented by using a high level packet capture library *libpcap*, or by using operating system specific mechanisms, such as *iptables* in Linux.

[1]When a network card is in promiscuous mode, it will receive all packets on the wire, even those destined to other hosts.

5.1.2 Intra-Island Encapsulation

The DM receives data packets from tunnels and multicast them into local island. This is not done by simply stripping off the encapsulation header and sending the application data to DATA_GROUP. Instead, the DM replaces the inter-island encapsulation header with an intra-island encapsulation header, send the packets to DATA_GROUP and UM_PORT, which is a UDP port reserved for UM and every Agent listens to. Upon receiving these packets, the Agent removes the intra-island header and sends the application data to local host's applications by setting TTL to 0 and destination to DATA_GROUP and application port.

The reason for taking an extra step in packet delivery is to carry some extra information in the intra-island header from the DM to other members in the island. Though the exact contents may vary as the protocol and implementation evolve and more applications are tested, there are basically three types of extra information.

End-to-End Though packets forwarded in UM may traverse many routers and hosts, the Agent should try to restore their end-to-end properties as much as possible, in order to keep the same service semantics and affect less to the applications. One example of such information is the original source address. When the DM multicast a packet to the local island, the packet's source address becomes the DM's, not the original source's. This will cause problems to some applications. A straightforward solution is to let DM change the source address to the original one, which is carried in the inter-island header. But this spoofed packet may be dropped in the network because it is likely to fail any address checking mechanism, especially the RPF checking used by many IP Multicast routing protocols. Therefore, the DM puts the original source address in the intra-island header and sends the packet without spoofing. When the receiver's Agent gets the packet, it can restore the source address to the original one and send it to applications on local host without worrying about address checking in the network.

Inter-island In multi-DM islands, packets multicast by one DM may be picked up by other DMs and forwarded further to other islands. Therefore information necessary for further inter-island routing should be carried with the packets in the intra-island header. The exact contents of such information depend on which protocol is used for inter-island routing. For example, HMTP does not require such information, but Narada needs the downstream DMs know the source island number so that they can apply RPF to decide whether to accept the packets or not, and CAN needs the downstream DMs know the incoming dimension number to decide where to forward the packets.

Intra-island Current island number is a piece of information that is only used for intra-island packet delivery. As discussed in Chapter 4, when a DM multicast packets into local island, it puts its current island number in the intra-island header. Upon receiving the packets, each Agent will check whether the packets are indeed from the same island that the receiver is in. If not, the Agent will discard the packets. Otherwise, it will forward them to applications on local host. Verifying current island is a simple and effective way to limit packet duplicates in transition period caused by network outages.

Without intra-island encapsulation, packets multicast into local island would contain only application data, lacking information like original source address and source island number. One way to carry the information without encapsulation is using IP option. But IP option can only have at most 40 bytes, which is too limited. Though at the expense of more processing overhead, intra-island encapsulation provides an effective way to carry various essential information from the DM to other members.

5.2 Group Directory

The Group Directory is a distributed directory service providing the mapping between GID and local multicast groups. The Head DM of each group periodically announces the group's mapping information to a well-known IP Multicast group, GD_GROUP, which every UM host in the island listens to and receives mapping information from. This soft-state approach gives the system adaptability and robustness needed to run on the Internet, where events like network change, packet loss, node failure may happen unexpectedly. It simplifies failure recovery as later arrived announcements will bring the system up-to-date no matter what faults happened. On the other hand, the soft-state approach incurs extra transmission overhead due to periodic announcements. When the number of groups is large, periodically sending all announcements will consume too much bandwidth. Many protocols like SAP [HPW00] slow down the sending rate such as $i = \frac{N*S}{R}$, where i is the interval between two consecutive announcements, N is the number of groups, S is the average size of groups, and R is a pre-configured rate limit. Though the bandwidth consumption is controlled under a rate limit, it takes hosts longer to get their directory information up to date. For example, on the current MBone, it takes about 30 minutes to receive all group announcements.

In UM, long synchronization time causes problems since the Agent checks the Group Directory first whenever it wants to join a group. If the Group Directory is not synchronized quickly on all UM hosts, multiple local groups may be created for one GID by different hosts. When an Agent starts, which may happen often because end hosts are turned on and off often, it has to endure a long start-up wait to collect all existing group announcements in the island. Caching previous directory information in hard disk helps shorten the start-up wait, but only to a limited extent when the directory information changes quickly or the time from last run is long. [SMR98] proposed to use a dedicated server as an announcement relay in local network. The relay server is supposed to be up all the time, collect all announcements, and re-send them to local network only with multicast scoping. Since the relay announcements are sent to local network, not the entire island, more bandwidth is likely available, and faster rate and shorter interval are possible. This approach has several drawbacks. First, it requires dedicated servers, which compromise the design's deployability. Second, the utilization of bandwidth is low because a constant high sending rate is used while most of the time a slow rate is enough for fault recovery. Third, the interval is still considered large when a new group is announced or a new UM host starts.

We adopt two techniques to speed up the synchronization of Group Directory while keep bandwidth usage low.

Staged Sending Rate When a group is announced for the first time, or its information has just been modified, it is considered a *hot* group and is announced in a very fast rate R_{hot}. Then the group enters the *warm* state where its sending rate is exponentially reduced, until a slow rate $R_{\text{cold}}$is reached and the group becomes *cold*. A cold group uses a fixed slow rate R_{cold}, and the transition from cold to hot is triggered by any change to the group information. This technique ensures important information (e.g., new groups, modified groups) are propagated quickly, while the overall bandwidth consumption is low because most of the groups are cold in most of the time.

Peer Help When a new UM host starts, in addition to listen to GD_GROUP, it looks for an existing UM host for help by using the multicast leader election technique. The newcomer multicast a HELP_REQUEST message to the GD_GROUP, and after a random delay one of existing UM hosts will multicast a HELP_REPLY message, which suppresses potential reply messages from other existing UM hosts. Then the newcomer opens a TCP connection to the helper fetching all group information the helper has collected, thus bringing the newcomer in sync with existing UM hosts. The new UM host conducts the search for helper in several rounds, starting from the smallest multicast scope and increasing the scope in each following round. With this expanding-ring search, newcomers will be helped by near-by UM hosts, which results in not only shorter transfer time for group information, but also less impact to remote hosts.

5.3 Implementation

We implemented a prototype of the UM Agent on Linux platform and tested it with some existing multicast applications. Fig. 5.1 shows the Agent's modules.

The Group Directory operates on a well-known IP Multicast address (GD_GROUP) and a well-known UDP port (GD_PORT). Group information is packed in SDP format, using the attribute lines ('a= ') to record UM-specific information like ASSERTION_GROUP, and sent as SAP packets. The Group Directory is implemented as a separate process from the Agent process, and it provides an interface via UNIX socket for local applications to query and modify the directory.

When the application wants to join a group, it makes a function call in libUM, which sends the join request to the Agent via a UNIX socket. The HGMP module will receive the join request, check the Group Directory, take corresponding

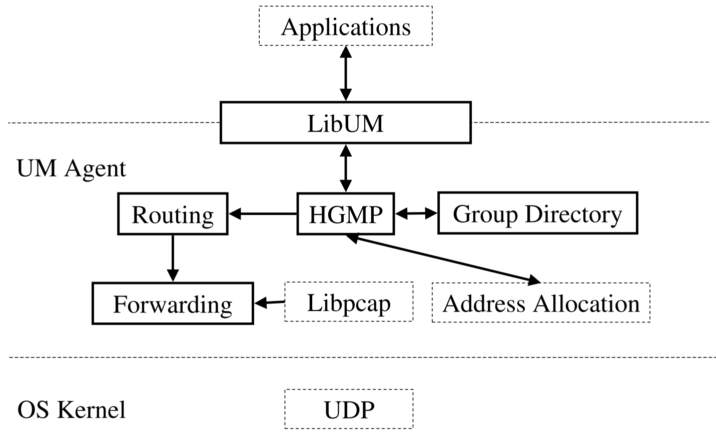


Figure 5.1: Modules in UM Agent

actions (see Chapter 4), and return the call. When the host becomes a DM of a group, the HGMP module will inform the routing module, which runs End-host Multicast protocol to build and maintain inter-island tunnels. Multiple End-host Multicast protocols can be implemented and linked in the routing module. For a particular group, which protocol to use is decided by the group owner when the group is created at its UMRP, and is disseminated via the Group Directory. The routing module maintains a routing table for the forwarding module, which receives data packets and forwards them according to the routing table. A simple multicast address allocation scheme, which picks an unused multicast address uniformly, is implemented. Libpcap is used for packet capture. All tunnel traffic and HGMP traffic use the same UDP port (UM_PORT) to ease administration work if firewall traversing is needed.

Applications need to call libUM functions when they join and leave a group.

int um_join(GID, DATA_GROUP, ...);
int um_leave(GID);

In um_join(), the application tells the Agent the group (GID) it wants to join, and the Agent returns the multicast address for local DATA_GROUP, to which the application should listen. Applications call um_leave() when they leave the group to help the Agent maintain a correct reference count to the group. However, since there is no guarantee that all applications will call um_leave(), the Agent also periodically checks the operating system's reference count of DATA_GROUP. If modifying source code is not possible for some existing IP Multicast applications, a wrapper program can be used to call um_join(), start the application using DATA_GROUP address as a command-line argument, and call um_leave() when the application finishes. Except these two function calls, writing UM programs is the same as writing IP Multicast programs.

We tested our Agent implementation with three different IP Multicast applications: sdr for best-effort multicast delivery, JMStudio for streaming media, and wb for reliable delivery. The testing network has two subnets, one is inside the MBone, and the other is outside. Each subnet has two hosts as group members. One host is set up as the UMRP for all tests. sdr [Han98] is a multicast session directory application that sends and listen session announcements in best-effort UDP packets. With UM, members outside of the MBone can run sdr and receive all session information currently being announced on the MBone. JMStudio [Sun03] is a java program that can transmit and receive streaming media over network via RTP/RTCP. This test reveals the necessity of using intra-island encapsulation to restore original source address. In RTP, each packet carries a 32-bit number called SSRC, which uniquely identifies the media stream. During a streaming session, without intra-island encapsulation, the receiver will associate the packet's SSRC with the packet's sender, the DM. When there is a DM change, the receiver will receive packets with the same SSRC but from a different source, the new DM. According to RTP specification [SCF96], the receiver will regard these packets as a new media stream with conflicting SSRC and drop them, though they actually belong to the old stream. In the test, every DM change is followed by a long pause at the player. Intra-island encapsulation restores the original source address for each packet, thus the receiver plays back the media smoothly. wb [LBN95] provides a shared white-board for users at different locations. It uses Scalable Reliable Multicast (SRM) [FJL97] to achieve reliable data delivery. It successfully runs on top of the multicast service provided by UM. All these applications are run within wrapper programs without modifying their source codes. Though primitive, these tests demonstrates that UM can support existing multicast applications and protocols with minimal efforts because it provides the same service model as IP Multicast's.

CHAPTER 6

Related Work

6.1 MBone Auto-Tunneling

The MBone was designed to facilitate the deployment of IP Multicast. It connects IP Multicast islands using tunnels, obviating IP multicast support on intermediate routers between the islands. Nevertheless, setting up the tunnel requires manual configuration and administrative privileges on routers at both ends, which makes connecting to the MBone an expensive proposition. Therefore, there are several proposals to automate the process of establishing tunnels to MBone.

UMTP UDP Multicast Tunnel Protocol (UMTP) [Fin01, FPR01] uses UDP tunnels between hosts, instead of IP-in-IP tunnels between routers, to connect individual hosts to the MBone. The choice of UDP tunnel makes it deployable on more operating systems, since some operating systems do not support IP-in-IP tunnels. When an isolated host wants to receive packets from a source inside the MBone, the host unicast a join request to the source. Along its path to the root, the join packet will be intercepted by the first multicast router it meets inside the MBone. Then the multicast router sends a reply back, establishing a UDP tunnel between them. Therefore, UMTP auto-tunneling supports on-demand UDP tunnels to the MBone for source-specific multicast (SSM) groups.

AMT Automatic Multicast Tunneling (AMT) [TTV01] deploys special servers or routers, called Relays if in the MBone, or Gateways if in isolated multicast islands. The Relays form a well-known anycast group, so that each Gateway can find a close Relay by sending unicast queries to the anycast group. When there is a request to join a group, the Gateway will forward an IGMP request to the Relay and establish on-demand tunnels to the MBone. AMT supports sourcing from the MBone, and sourcing SSM traffic from the isolated site. It does not support sourcing general IP Multicast traffic from the isolated site.

CastGate Castgate [Lie01] builds a DNS-like hierarchical, distributed database to support MBone tunnel management. All potential tunnel end-points within the MBone must register themselves in the database. To join the MBone, a host first looks up the database for a close end point with available bandwidth, then establishes a tunnel to that end point.

Existing MBone auto-tunneling proposals only automate the setup of static tunnels. They usually do not support dynamic re-configuration after tunnels are established. But most importantly, they all require infrastructure support, either dedicated servers or routers, which continues to undermine the deployability of MBone.

6.2 End-host Multicast

End-host Multicast has drawn much attention recently in the research community as many protocols and designs have emerged. End-host Multicast has minimal deployment barriers because the only requirement is for the users to install a program on their own computers. End-host Multicast protocols differ in their target applications and routing algorithms. Judging by the type of data delivery tree, they can be categorized as shared-tree protocols and source-tree protocols. Generally speaking, shared-tree protocols maintain less state information at each node, thus more scalable; While source-tree protocols utilize more overlay links, thus able to achieve shorter delay.

6.2.1 Shared-Tree Protocols

ALMI Application Layer Multicast Infrastructure (ALMI) [PSV01] targets at applications with small group size. It takes a centralized approach to build the shared-tree. Group members measures the delay between them and report the measurement results to a central controller. The controller computes a minimum spanning tree based on these measurements and disseminates the resulting routing tables to all members. Though the controller has the potential to calculate a tree structure to optimize application performance, the design's centralized nature limits its scalability and robustness.

Yoid Your Own Internet Distribution (Yoid) [Fra01] is a set of protocols forming a new architecture for general host-based content distribution, including multicast. Yoid consists of three major parts: the Yoid Identification Protocol (YIDP) provides a foundation for host addressing within the context of a given group; the Yoid Distribution Protocols (YDP) are a family of transport protocols including realtime, non-realtime, reliable, and non-reliable functionality; the Yoid Tree Management Protocol (YTMP) is responsible for tree construction and management. In YTMP, the rendezvous host maintains a list of some current members. A new member obtains the list by contacting the rendezvous, then from the list chooses a member with spare fan-out as its parent. Over time a member does a random walk on the tree, measures delays to the members it encounters, and switches parents if it finds a nearer one. YTMP adopts loop prevention algorithms, which requires coordination among multiple members involved in one potential parent switch. YTMP also maintains a mesh structure among group members. The mesh is used to increase robustness, e.g., when the tree is partitioned, packets can be delivered via the mesh. Compared with Yoid, UM provides only best-effort IP Multicast delivery service, not general purpose content delivery. The HMTP protocol uses a more orderly join algorithm, thus converges faster and usually results in shorter delay. HMTP does not have the mesh structure, or the loop prevention operations. Yoid utilizes local IP Multicast connectivity by selecting one member as the cluster head, similar to the designated member in UM. But how to select the cluster head and how to support multiple heads per cluster are not specified.

BTP Banana Tree Protocol (BTP) [HJ02] is the End-host Multicast protocol used by Jungle Monkey, a distributed file sharing application. In BTP, a member joins a group by connecting to the root of the tree. It then looks for closer parent nodes to switch to over time. A member can only switch to a nearby sibling or the grandparent node. To prevent possible loops, BTP adopts some conservative rules for a node to decide whether to accept a join request or not.

HostCast HostCast [LM03] assumes the multicast group has a single source, the root of the shared-tree. A new member looks for parent by searching the tree in a width-first manner, and joins the first node with available out-degree it finds. HostCast also maintains a mesh structure, consisting of all links between a node and its parent, its grandparent, and its uncles. The root periodically sends small fixed-size packets to the mesh, so that members can measure their path QoS from the root, and switch parent if necessary. The mesh structure also helps recover the primary tree quickly in case of partition. HostCast cannot be easily extended to any-source multicast groups, since its join algorithm, QoS measurements, and switch algorithm all depend on the single-source assumption.

TAG Topology Aware Grouping (TAG) [KF02] exploits the underlying network topology information to build the multicast tree. The root discovers the network path to group members using several tools like traceroute. When a new member joins, it selects its parent node as the one whose network path from the root has maximal overlap with the new member's network path from the root. A data source unicast packets to the root first, then the root will forward the packets down to the tree. Therefore, TAG builds a uni-directional shared-tree congruent to the network topology. Since the path measurement usually takes time, and may even be blocked by firewalls, adapting to network conditions is hard for TAG. The uni-directional tree also makes it best suited for single-source applications only.

SALM Scalable Application Layer Multicast (SALM) [BBK02] organizes group members into a hierarchical structure. At each leave of the hierarchy, there is one or more "clusters", and each cluster has a leader. A leader is elected to be the host which has the minimum maximum distance to all other hosts in the cluster. A cluster size has lower and upper bounds. The leader periodically checks its cluster size, and splits or merges the cluster if necessary. Leaders at the same level form clusters at a higher level, until the highest level where only the root resides. A new member joins the tree by searching the hierarchy in a depth-first manner, similar to HMTP join algorithm. The hierarchy adapts to membership changes as the clusters may split and merge, and cluster heads may come and go. In packet forwarding, SALM uses a delegation technique to avoid overloading individual nodes. The hierarchy and cluster size constraint help reduce the depth of the tree, but maintaining it makes the protocol more complex.

6.2.2 Source-Tree Protocols

Narada/ESM End System Multicast (ESM) [CRZ00, CR01] envisions that a large number of small-sized groups will be dominant in future Internet, therefore multicast functionality should be implemented at end hosts only against to routers. Narada is a mesh-based protocol used in ESM. A new member joins the mesh by connecting to a randomly chosen existing member. Then the new node learns all the group members from its neighbor and starts measuring delay and bandwidth to other members. Each member periodically evaluates every possible overlay link from others to itself. An overlay link is added if the gain exceeds a threshold, or removed if its utility drops below a threshold. Based on this dynamically adjusted mesh structure, a DVMRP-like protocol is run by every member to generate source-trees for packet forwarding. Narada is able to utilize the potential rich connectivity of the overlay. But since each member has to maintain a full list of all members, its scalability is limited to small groups only.

ALM-CAN ALM-CAN [RHK01] is an End-host Multicast protocol built on top of Content Addressable Network (CAN), a peer-to-peer network providing dynamic hashing table functions. Nodes in CAN form a multi-dimensional Cartesian coordinate space. Every node in the CAN owns a portion of the total space, called a zone, and point-to-point message in CAN is routed along the shortest path from a source to its destination in the Cartesian coordinate space. In multicast, the source sends the packet to all its neighbors. Upon receiving a multicast packet along dimension k, a node forwards the packet along all dimensions less than or equal to k. A node does not forward a packet along a dimension if that packet has already traversed half-way from the source along that dimension. These forwarding rules minimize duplicate delivery of multicast packets.

TMesh TMesh [WHJ02] improves shared-tree protocols by adding shortcut overlay links, thus change the tree into a mesh structure. Simulations have shown that for well constructed shared-tree like HMTP tree, adding a few shortcuts can significantly reduce the average delay ratio. TMesh evaluates an overlay link by using some local heuristics, including calculating the improvement of delay ratio and a utility function. Simulation results show that TMesh with HMTP can achieve average delay ratio 1.3 to 2.5 times of IP Multicast delay. On top of the mesh, TMesh runs DVMRP to obtain source-trees for packet forwarding.

6.3 Multicast Server Overlay

Compared with IP Multicast, pure End-host Multicast cannot achieve the same performance. Especially for large groups, the penalty incurred by the duplicate transmission and inefficient routing becomes significant. Multicast server overlay is an approach which places a large number of dedicated servers on the Internet strategically to provide the multicast service for end users. Its performance is expected to be better since servers tend to be more powerful and have more bandwidth; but its deployability is compromised since the service depends on the success of the server overlay's deployment by a third-party.

Scattercast The Scattercast architecture is intended for wide-area content distribution [Cha00]. Its servers, called SCXs, are strategically placed worldwide and self-organized into an overlay mesh using the Gossamer protocol. Gossamer is similar to Narada, where member maintains a complete list of all members, continuously measures delay to others, and evaluate every possible overlay link, but it uses a different utility function in evaluating an overlay link. Using either static configuration or DNS redirection, client hosts locate the closest SCX to connect. If configured, local IP Multicast can be utilized. Sources announce their intent to send data to their SCXs, and SCXs runs DVMRP to build reverse shortest paths over the mesh structure for the source. The scalability problem of the Gossamer protocol is a limiting factor to how many SCXs can be deployed.

Overcast The Overcast [JGJ00] system consists of a single source, a number of Overcast nodes as servers in the network, and client hosts. Overcast provides reliable single-source multicast over the shared-tree rooted at the source. Unlike most other End-host Multicast protocols, Overcast uses HTTP as the transport protocol, and bandwidth as the routing metric. Overcast tries to maximize the bandwidth from the root to each member. Thus it is best suited for applications like Internet multimedia streaming. A new Overcast node joins the tree starting at root. It tries to move down on the tree without losing significant bandwidth to the root node. Periodically, each node measures its bandwidth to its current siblings, parent, and grand-parent by downloading 10 Kbytes data. Based on the results, the node may decide to switch to another parent. Client hosts connect to an Overcast node and receive data via HTTP.

AMcast In AMcast [ST02b, ST02a, STW01], the dedicated servers are called Multicast Service Nodes (MSN). These nodes are deployed on the Internet by a single service provider, who has complete knowledge of the entire MSN overlay. Therefore AMcast uses centralized approach to determine where to deploy MSNs, how to connect them into an overlay, and how to route packets given traffic estimates. These problems are abstracted to the construction of minimum-diameter degree-limited spanning tree and limited diameter residual-balanced spanning tree, and heuristic solutions are proposed.

OMNI Overlay Multicast Network Infrastructure [BKK03] is intended for latency-sensitive real-time applications, such as multimedia streaming. OMNI dynamically adapts the multicast tree based on the relative importance of each MSN, which is based on the number of its clients. OMNI attempts to minimize the average latency observed by all clients. There is a maximum fan-out for each MSN node. OMNI protocol has an initialization phase followed by a series of incremental refinements. In the initialization phase, a tree is created by the source MSN, based only on the latencies between it and all other MSNs. In each refinement step, a small number of nearby MSNs probe each other, and switch parent if necessary. An MSN node also randomly walks the entire tree looking for better parent. OMNI does not specify how clients discover an MSN, or whether native IP Multicast can be utilized.

Broadcast Federation Broadcast Federation [CS02] is an architecture that enables the composition of different broadcast-capable networks to provide an Internet-wide broadcast service. Each Broadcast Network (BN) may have its own broadcast mechanism, such as native IP Multicast, SSM, and unicast, but they are inter-connected by peering between Broadcast Gateways (BG), which are special servers in each BN. It uses a path-vector protocol to propagate reachability information to all BGs, a shared-tree protocol to build data dissemination tree, and a forwarding protocol to deliver packets between BNs. In Broadcast Federation, the BGs and the peering between BNs are administratively configured at each BN. While in UM, DMs are dynamically elected from normal members, and the inter-island tunnels are dynamically maintained.

CHAPTER 7

Conclusion

7.1 Summary of Accomplishments

In this dissertation research, we designed and implemented the Universal Multicast (UM) framework to provide ubiquitous multicast delivery service on the Internet. As a basic service common to most group communication applications, multicast should be implemented as part of the network infrastructure for the sake of performance and scalability. On the other hand, from the deployment point of view, the path with least resistance is evolving from network edges towards the core. UM solves this dilemma by having a flexible design that allows the evolution of multicast deployment. Unlike previous work, UM does not implement multicast functionality at a fixed place in the network for a particular deployment stage; instead, it is a general framework within which applications can be provided with IP Multicast delivery immediately, and infrastructure support for multicast can gradually spread from the network edges into the core.

We designed HMTP, an End-host Multicast protocol, as a basic inter-island routing protocol for UM. HMTP builds a shared tree to connect all participating islands. New members probe a small part of the tree to find a close member as the parent to join. Each member keeps state information of its local neighbors and only a few remote members. The tree is maintained by soft-state refresh messages. It adapts to changes in network condition and group membership. It also has the ability to detect and break routing loops, and recover from network partition. Simulation results show that HMTP tree converges fast, has low tree cost, moderate delay, and low network link load. This simple, scalable shared-tree protocol is designed for general multicast applications. Follow-up work demonstrates that HMTP can be used as the foundation for more complex protocols targeted at particular applications (e.g., delay sensitive application).

HGMP glues inter-island End-host Multicast and intra-island IP Multicast together. It expands each node in the End-host Multicast protocol to encompass an IP-Multicast-enabled island. It is fully distributed and self-organized without any dependence on routers or dedicated servers. Nevertheless, HGMP can automatically make use of infrastructure support in a way transparent to applications. It supports multiple DMs per island to reduce delivery latency and traffic concentration, especially for large islands. HGMP is protocol-independent, which means the design can work with any IP Multicast routing protocol and any End-host Multicast protocol. HGMP is robust against network changes in that the damage caused by network partition or merging is minimized.

We implemented the UM framework as UM Agent, a user-space daemon program providing all UM functionality at every group member. In UM, packet forwarding is alternately done by hosts (between islands) and routers (within islands), which raises some issues in implementation. UM Agent uses packet capture to receive multicast packets originated from local island. It also uses intra-island encapsulation to carry extra information, such as original source address, from the DM to normal members. Since Group Directory plays an important role in HGMP operation, we use staged sending rate and peer help to shorten its start-up time within bandwidth limit. Tests with three different multicast applications demonstrated that UM can bring ubiquitous multicast connectivity to existing applications and protocols with minimal efforts.

7.2 Future Work

A challenging future work is fault-tolerant data delivery. Usually end hosts are more prone to faults, either accidentally or intentionally, than routers. Therefore fault-tolerant data delivery becomes more important in overlay networks that use end hosts to forward packets. The multicast service can be disrupted or negatively affected by misbehaving hosts, which could be caused by malicious attack, software bugs, or even hardware glitches. Cryptographic solutions are often not enough, partly because a public key infrastructure is not available in such environments. In a multicast overlay, since every member is supposed to receive the same packets eventually, one member can check with others to see if its upstream neighbor behaves correctly in forwarding packets. Issues like how to record and compare packets, how to identify misbehavior and how to repair faults etc. need to be studied. Such a fault-tolerant scheme would be an add-on to existing End-host Multicast protocols. This direction can extend to general peer-to-peer applications, where centralized management is not available and an ad hoc trust model is needed for fault-tolerance and security.

The UM framework also needs more work and practical experiences to improve. For example, experiences from real usage can help decide which assertion scope is good for which network environment. Implementing more End-host Multicast protocols will take advantage of UM's protocol-independence and make the Agent suitable for more applications. Further tests with other existing applications and upper layer protocols are needed in order to improve UM's compatibility with IP Multicast. Larger scale experimentations in real world and standardization efforts in IETF can help spur multicast application development with UM.

References

[BBK02] S. Banerjee, B. Bhattacharjee, and C. Kommareddy. "Scalable Application Layer Multicast." In *Proc. of ACM SIGCOMM*, September 2002.

[BC88] T. Bates, R. Chandra, , D. Katz, and Y. Rekhter. "Multiprotocol extensions for BGP-4." RFC 2283, IETF, February 1988.

[BFC93] T. Ballardie, P. Francis, and J. Crowcroft. "Core based trees CBT." In *Proc. of ACM SIGCOMM*, pp. 85–95, September 1993.

[BKK03] S. Banerjee, C. Kommareddy, K. Kar, B. Bhattacharjee, and S. Khuller. "Construction of an Efficient Overlay Multicast Infrastructure for Real-time Applications." In *Proc. of IEEE INFOCOM*, April 2003.

[CA01] R. Chalmers and K. Almeroth. "Modeling the Branching Characteristics and Efficiency Gains of Global Multicast Trees." In *Proc. of IEEE INFOCOM*, pp. 449–458, 2001.

[Cas95] S. Casner. "MBone Topology." May 1995.

[CCG02] Q. Chen, H. Chang, R. Govindan, S. Jamin, S. Shenker, and W. Willinger. "The Origin of Power-Laws in Internet Topologies Revisited." In *IEEE INFOCOM*, June 2002.

[CD92] S. Casner and S. Deering. "First IETF Internet Audiocast." *ACM Computer Communication Review*, **22**(3):92–97, July 1992.

[CFD01] L. H. M. K. Costa, S. Fdida, and O. C. M. B. Duarte. "Hop by hop multicast routing protocol." In *Proc. of ACM SIGCOMM*, September 2001.

[Cha00] Y. Chawathe. *Scattercast: an architecture for Internet broadcast distribution as an infrastructure service.* PhD thesis, University of California, Berkeley, December 2000.

[CR01] Y. Chu, S. G. Rao, , S. Seshan, and H. Zhang. "Enabling conferencing applications on the Internet using an overlay multicast architecture." In *Proc. of ACM SIGCOMM*, August 2001.

[CRZ00] Y. Chu, S. G. Rao, and H. Zhang. "A case for end system multicast." In *Proc. of ACM SIGMETRICS*, pp. 1–12, June 2000.

[CS01] J. Chuang and M. Sirbu. "Pricing Multicast Communication: A Cost-Based Approach." *Telecommunication Systems*, **17**(3):281–297, 2001.

[CS02] Y. Chawathe and M. Seshadri. "Broadcast Federation: an application-layer broadcast internetwork." In *Proc. of the Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, May 2002.

[DC90] S. Deering and D. Cheriton. "Multicast routing in datagram internetworks and extended LANs." *ACM Transactions on Computer Systems*, **8**(2):85–110, May 1990.

[DLL00] C. Diot, B. N. Levine, B. Liles, H. Kassem, and D. Balensiefen. "Deployment issues for the IP multicast service and architecture." *IEEE Network Magazine*, pp. 78–88, January 2000.

[EFH98] D. Estrin, D. Farinacci, A. Helmy, D. Thaler, S. Deering, M. Handley, V. Jacobson, C. Liu, P. Sharma, and L. Wei. "Protocol Independent Multicast-Sparse Mode (PIM-SM): protocol specification." RFC 2362, IETF, June 1998.

[Eri94] H. Eriksson. "MBONE: the multicast backbone." In *Communications of the ACM*, August 1994.

[Eub03] M. Eubanks. "Multicast Status Web Page, Multicast Technologies, Inc." http://www.multicasttech.com/status/, April 2003.

[FCG01] A. Fei, J. Cui, M. Gerla, and M. Faloutsos. "Aggregated Multicast: an Approach to Reduce Multicast State." In *Proc. of 6th Global Internet Symposium (GI2001) in conjunction with Globecom*, November 2001.

[Fen97] W. Fenner. "Internet group management protocol, Version 2." RFC 2236, IETF, November 1997.

[Fin01] R. Finlayson. "The UDP multicast tunneling protocol." Internet Draft, IETF, March 2001.

[FJJ01] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. "IDMaps: A global Internet host distance estimation service." *ACM/IEEE Transactions on Networking*, October 2001.

[FJL97] S. Floyd, V. Jacobson, C. Liu, S. McCanne, and L. Zhang. "A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing." *ACM/IEEE Transactions on Networking*, **5**(6):784–803, 1997.

[FPR01] R. Finlayson, R. Perlman, and D. Rajwan. "Accelerating the deployment of multicast using automatic tunneling." Internet Draft, IETF, February 2001.

[Fra01] P. Francis. "Yoid: your own Internet distribution." http://www.isi.edu/div7/yoid/, March 2001.

[Han98] M. Handley. "Session directories and scalable Internet multicast address allocation." *ACM Computer Communication Review*, **28**(4):105–116, September 1998.

[HC00] H. Holbrook and B. Cain. "Source-specific multicast for IP." Internet Draft, IETF, November 2000.

[HH00] M. Handley and S. Hanna. "Multicast Address Allocation Protocol (AAP)." Internet Draft, IETF, December 2000.

[HJ02] D. Helder and S. Jamin. "End-host multicast communication using switch-trees protocols." In *Global and Peer-to-Peer Computing on Large Scale Distributed Systems (GP2PC)*, May 2002.

[HPS99] S. Hanna, B. Patel, and M. Shah. "Multicast Address Dynamic Client Allocation Protocol (MADCAP)." RFC 2730, IETF, December 1999.

[HPW00] M. Handley, C. Perkins, and E. Whelan. "Session Announcement Protocol." RFC 2974, IETF, October 2000.

[HTK00] M. Handley, D. Thaler, and R. Kermode. "Multicast-Scope Zone Announcement Protocol (MZAP)." RFC 2776, IETF, February 2000.

[JGJ00] J. Jannotti, D. Gifford, K. Johnson, M. Kaashoek, and J. O'Toole. "Overcast: reliable multicasting with an overlay network." In *Proc. of the Symposium on Operating Systems Design and Implementation*, October 2000.

[Kah95] J. Kahn. "MBONE – Communications for the Next Millennium." http://www.lbl.gov/ICSD/MBONE/, July 1995.

[KF02] M. Kwon and S. Fahmy. "Topology-aware overlay networks for group communication." In *Proc. of the Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, pp. 127–136, May 2002.

[KRT98] S. Kumar, P. Radoslavov, D. Thaler, C. Alaettinoglu, D. Estrin, and M. Handley. "The MASC/BGMP architecture for inter-domain multicast routing." In *Proc. of ACM SIGCOMM*, pp. 93–104, 1998.

[LBN95] LBNL's Network Research Group. "wb: the shared whiteboard." ftp://ftp.ee.lbl.gov/conferencing/wb/, 1995.

[Lie01] P. Liefooghe. "CastGate: an auto-tunneling architecture for IP multicast." Internet Draft, IETF, November 2001.

[LM03] Z. Li and P. Mohapatra. "HostCast: A New Overlay Multicast Routing Protocol." In *IEEE Int'l Conference on Communications*, May 2003.

[Mey98] D. Meyer. "Administratively Scoped IP Multicast." RFC 2365, IETF, July 1998.

[MF01] D. Meyer and W. Fenner. "Multicast source discovery protocol (MSDP)." Internet Draft, IETF, May 2001.

[ML01] D. Meyer and P. Lothberg. "GLOP Addressing in 233/8." RFC 3180, IETF, September 2001.

[Moy94] J. Moy. "Multicast extensions to OSPF." RFC 1584, IETF, March 1994.

[Nat] National Laboratory of Applied Network Research, NSF Cooperative Agreement No. ANI-9807479. "NLANR active measurement project." http://watt.nlanr.net/.

[PLB99] 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, IETF, March 1999.

[PSS98] P. Parnes, K. Synnes, and D. Schefström. "Lightweight application level multicast tunneling using mTunnel." In *Computer Communications*, 1998.

[PSV01] D. Pendarakis, S. Shi, D. Verma, and M. Waldvogel. "ALMI: an application level multicast infrastructure." In *Proceedings of 3rd Usenix Symposium on Internet Technologies and Systems (USITS 2001)*, March 2001.

[REG00] P. Radoslavov, D. Estrin, R. Govindan, M. Handley, S. Kumar, and D. Thaler. "The Multicast Address-Set Claim (MASC) Protocol." RFC 2900, IETF, September 2000.

[RHK01] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. "Application-Level Multicast Using Content-Addressable Networks." In *Networked Group Communication*, pp. 14–29, 2001.

[SCF96] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. "RTP: A Transport Protocol for Real-Time Applications." RFC 1889, IETF, January 1996.

[Sch00] E. Schooler. *Why Multicast Protocols (Don't) Scale: An Analysis of Multipoint Algorithms for Scalable Group Communication*. PhD thesis, California Institute of Technology, September 2000.

[SMR98] A. Swan, S. McCanne, and L. Rowe. "Layered Transmission and Caching for the Multicast Session Directory Service." In *ACM Multimedia*, September 1998.

[SNZ00] I. Stoica, T. S. E. Ng, and H. Zhang. "REUNITE: a recursive unicast approach to multicast." In *Proc. of IEEE INFOCOM 2000*, March 2000.

[SRC84] J. Saltzer, D. Reed, and D. Clark. "End-To-End Arguments in System Design." *ACM Transactions on Computer Systems*, **2**(4):277–288, November 1984.

[ST02a] S. Shi and J. Turner. "Placing Servers in Overlay Networks." In *Proc. of International Symposium on Perform ance Evaluation of Computer and Telecommunication Systems (SPETS)*, July 2002.

[ST02b] S. Shi and J. Turner. "Routing in Overlay Multicast Networks." In *Proc. of IEEE INFOCOM*, June 2002.

[STW01] S. Shi, J. Turner, and M. Waldvogel. "Dimensioning Server Access Bandwidth and Multicast Routing in Overlay Networks." In *Proc. of the Int'l Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, June 2001.

[Sun03] Sun Microsystems, Inc. "Java Media Framework APIs." http://java.sun.com/products/java-media/jmf/, 2003.

[THE00] D. Thaler, M. Handley, and D. Estrin. "The Internet multicast address allocation architecture." RFC 2908, IETF, September 2000.

[TTV01] D. Thaler, M. Talwar, L. Vicisano, and D. Ooms. "IPv4 automatic multicast without explicit tunnels (AMT)." Internet Draft, IETF, February 2001.

[WE94] L. Wei and D. Estrin. "A comparison of multicast trees and algorithms." In *Proc. of IEEE INFOCOM*, June 1994.

[WHJ02] W. Wang, D. Helder, S. Jamin, and L. Zhang. "Overlay Optimizations for End-host Multicast." In *Proc. of the Int'l Workshop on Networked Group Communication (NGC)*, October 2002.

[WPD88] D. Waitzman, C. Partridge, and S. Deering. "Distance vector multicast routing protocol." RFC 1075, IETF, November 1988.

[ZCB96] E. Zegura, K. Calvert, and S. Bhattacharjee. "How to model an internetwork." In *Proc. of IEEE INFOCOM*, March 1996.

[ZJZ02] B. Zhang, S. Jamin, and L. Zhang. "Host Multicast: a framework for delivering multicast to end users." In *Proc. of IEEE INFOCOM*, June 2002.