December 1991

Report No. STAN-CS-92-1415

*Thesis*

PB96-148721

# Multicast Routing in a Datagram Internetwork

by

**Stephen Edward Deering**



**Department of Computer Science**

**Stanford University**

**Stanford, California 94305**

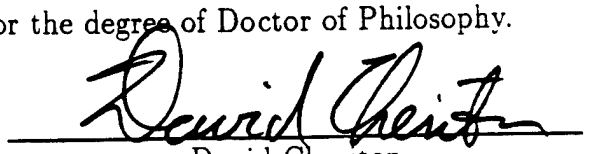


# MULTICAST ROUTING
# IN A DATAGRAM INTERNETWORK

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF ELECTRICAL ENGINEERING
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

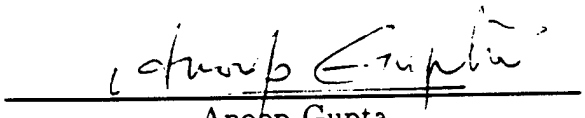
By
Stephen Edward Deering
December 1991

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



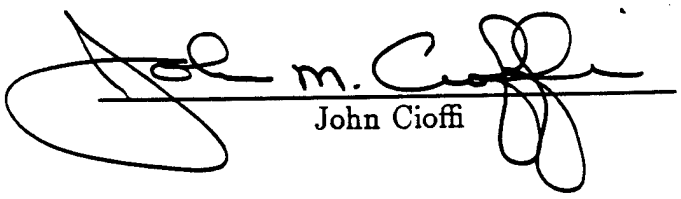
David Cheriton
(Principal Advisor)

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



Anopp Gupta

I certify that I have read this dissertation and that in my opinion it is fully adequate, in scope and in quality, as a dissertation for the degree of Doctor of Philosophy.



John Cioffi

Approved for the University Committee
on Graduate Studies:

# Abstract

Most local-area networks, as well as some other packet-switched networks, support *multicast*, the ability to address and deliver a packet to a *set* of destinations. Currently, when those networks are interconnected by routers or bridges to form an *internetwork*, the multicast service is either unavailable beyond a single network—as when using IP routers—or is offered in a way that significantly limits the scalability of the internetwork—as when using LAN bridges. To address those problems, we present a new service model for multicasting in a datagram (or *connectionless*) internetwork, and a set of new store-and-forward multicast routing algorithms to support that service model.

The multicast service model, which we call the Host Group Model, is a natural generalization of the *unicast* service model offered by datagram internetworks. Multicast packets are delivered to each member of a multicast group with the same "best-efforts" reliability and performance as unicast packets to that member. Multicast groups may be of arbitrary size, may change membership dynamically, and may have either global scope, that is, with members located anywhere in the internetwork, or local scope, with members confined to a particular administrative domain. Senders of multicast packets need not belong to the destination groups and need not know the membership of the groups.

The new multicast routing algorithms to support the Host Group Model take the form of extensions to the two unicast routing algorithms most commonly used in network-layer routers—the distance-vector algorithm and the link-state algorithm—and to the spanning-tree algorithm used by most datalink-layer bridges. In all cases, the delivery path of a multicast packet forms a tree, rooted at the sender, with copies

of the packet being generated only at those points where the tree branches. The routing algorithms have low overhead, good performance, and scalability as good as, or better than, unicast routing. They may be used hierarchically, alone or in combination, to support multicasting across very large-scale internetworks.

# Acknowledgements

The work presented in this dissertation was initially suggested, continually supported, and frequently inspired by my advisor, David Cheriton. I am deeply grateful for his guidance, his perseverance, and his friendship.

I am also indebted to David for enabling me to participate in the Internet End-to-End Protocols Research Group. The group, chaired by Bob Braden, has encouraged and supported this work and has made it possible for some of the results of this work to be put into practice in the DARPA Internet.

I benefited greatly from the advice and insights of the members, present and former, of my Dissertation and Oral Examination Committees—Anoop Gupta, John Cioffi, Mark Linton, Flaviu Cristian, and Von Eshleman. I owe many thanks to my peers at Stanford—Cary Gray, Bruce Hitson, Hemant Kanakia, Tim Mann, Jeff Mogul, Joe Pallas, Marvin Theimer, and the rest of the Distributed Systems Group—and to my colleagues in the Internet community—Scott Brim, Karen Lam, John Moy, Craig Partridge, and David Waitzman—for suggesting, critiquing, and helping to refine many of the ideas presented here. Thanks also to Dan Swinehart for reviewing the final copy of this document.

I gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada, the US Defense Advance Research Projects Agency under contracts N00039-83-K-0431, N00039-84-C-0211, and N00039-86-K-0431, the US National Science Foundation under grant DCR-83-52048, and Digital Equipment Corporation. I would also like to thank the Xerox Corporation, my current employer, for allowing me to spend part of my paid time completing this dissertation.

And I would especially like to thank my wife, Sharon Brunzel, and the rest of my

friends and family for being there when I needed them, and for knowing when not to ask, "How's your thesis going?"

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

A packet-switched network is said to provide a *multicast* service if it can deliver a packet to a *set* of destinations, rather than just a single destination. Most local-area networks, such as Ethernet [28] and Token Ring [38], provide a multicast service, and that service has been exploited by many applications and distributed systems. Unfortunately, when those networks are interconnected to form an *internetwork*, the multicast service is either:

- not offered beyond a single network, as is the case when the networks are connected by network-layer[1] routers such as DARPA IP Gateways [13] or ISO Intermediate Systems [45], or
- offered in a way that significantly limits the scalability of the internetwork, as is the case when the networks are connected by datalink-layer bridges such as the DEC LANBridge [35] or the Vitalink TransLAN [34].

This dissertation presents new algorithms for such routers and bridges that overcome those limitations, to provide a scalable, internetwork-wide multicast service.

The type of multicast service addressed in this work is *datagram* (or *connectionless*) service, as is the service provided by most local-area networks and by internetwork protocols like DARPA IP [57] and ISO CLNP [46]. The new algorithms take the form of extensions to the two network-layer routing algorithms most commonly

[1] using the layer terminology of the ISO Model for Open Systems Interconnection [41].

used in datagram internetworks—the distance-vector algorithm and the link-state algorithm—and to the spanning-tree algorithm used by most datalink-layer bridges. Like the base algorithms, the extended algorithms can be combined hierarchically, to provide multicast service across large, heterogeneous internetworks.

This introductory chapter identifies the benefits of multicasting and its potential value as an internetwork service. It also introduces some basic terminology used throughout the dissertation, and relates it to an abstract architectural model of an internetwork and to some concrete examples. The chapter finishes with an overview of the rest of the dissertation.

## 1.1 Benefits of Multicasting

A multicast service can offer two important benefits to network applications:

**Efficient multi-destination delivery.** When an application must send the same information to more than one destination, multicasting is more efficient than *unicasting* separate copies to each destination—it reduces the transmission overhead on the sender and, depending on how it is implemented, it can reduce the overhead on the network and the time taken for all destinations to receive the information. Examples of applications that can take advantage of multi-destination delivery are:

- updating all copies of a replicated file or database [14, 16].
- sending voice, video, or data packets to all participants in a computer-mediated conference [59].
- disseminating intermediate results to a set of processors supporting a distributed computation [19].

**Robust unknown-destination delivery.** If a set of destinations can be identified by a single *group address* (rather than, say, a list of individual addresses), such a group address can be used to reach one or more destinations whose individual addresses are unknown to the sender, or whose addresses may change over time.

Sometimes called *logical addressing* or *location-independent addressing*, this use of multicast serves as a simple, robust alternative to configuration files, directory servers, or other binding mechanisms. Examples of applications that can take advantage of logical addressing are:

- querying a distributed database or file store, where the particular location of the desired data is unknown [48].
- locating an instance of a particular network service, such as name service [52] or bootstrap service [23].
- sending sensor readings or status reports to a self-selected, changeable set of monitoring stations [15]. (This is an example of an application that exploits both the logical addressing and the multi-destination delivery properties of multicast.)

Because of these benefits, multicast has seen widespread use in those networks that support it, primarily local-area networks, but also satellite [30] and terrestrial packet radio [21] networks. Access to the multicast capability has been provided by popular protocol suites, such as Sun's broadcast RPC service [63][2] and IBM's NetBIOS [40], and by distributed programming systems such as the V System [20] and the Andrew distributed computing environment [60]. In some cases, multicast has played an important role in organizing the underlying protocols and operating systems themselves, as well as being offered as a service for applications.

Extending the multicast service across an internetwork would have the following advantages:

[2]In some environments, the multicast service has (unfortunately) been based on a network's *broadcast* addressing facility, which delivers packets to *all* destinations in the network and requires software at each destination to recognize and discard unwanted packets. While such an implementation of multicast does provide the benefits of multi-destination delivery and logical addressing, it incurs undesirable overhead in all those destinations that must process unwanted packets, overhead that gets worse as more and more applications use multicast. Fortunately, this problem can be avoided in modern local-area networks, such as Ethernet and other networks conforming to the IEEE 802 standards [37], that provide a large space of multicast addresses that can be recognized and filtered by network interface hardware.

- It would ease the migration of existing multicast-based applications and distributed systems to an internetwork environment. Currently, to adapt such applications to an internetwork requires either:
  - rewriting them to use unicast only, giving up the efficiency of multicast and replacing the simple binding capability of multicast with more complicated or fragile mechanisms, or
  - implementing special-purpose servers to simulate an internetwork multicast service, as is being done for NetBIOS over IP [1]. Without the assistance of the internetwork routers, such servers cannot offer the same efficiency or ease-of-configuration as a true internetwork multicast service.
- It would allow existing internetwork applications to become more efficient and more robust. For example, protocols for handling electronic mailing lists or network news distribution might be made much more efficient by the use of a multicast service. The logical addressing capability of multicast could be used to replace the manually-maintained configuration files currently used for locating internetwork services, such as directory servers and time servers. As it is, the lack of a general internetwork multicast service has, in some cases, led to *ad hoc* router extensions to support single-purpose, limited multicasting, for example, to support access to remote bootstrap services [23].
- It would allow new classes of applications to be supported by existing datagram internetworks, such as teleconferencing or video distribution which require concurrent, real-time delivery of data to multiple destinations. A general-purpose internetwork multicast service might eliminate or reduce the need for alternative, special-purpose routing services, such as that provided by the DARPA Stream Protocol [32] (a routing protocol implemented in some DARPA Internet gateways to support real-time, multi-participant conferencing).

Given the potential value of an internetwork multicast service, the questions arise: what should the service look like, how much would it cost in terms of network resources, and how well would it scale? These are the questions addressed by this

dissertation.

## 1.2 Architectural Model and Terminology

The term *internetwork* is used in this work to refer to any packet forwarding system that can be modeled as illustrated in Figure 1.1. There are a variety of existing

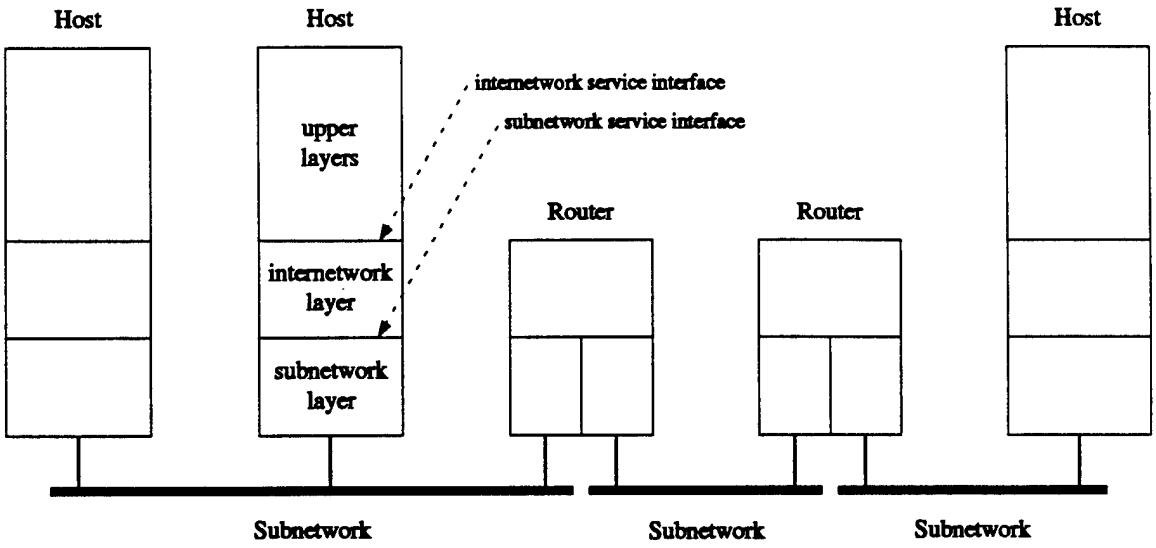



Figure 1.1: Architectural Model of an Internetwork

systems that fit this abstract model, including not only those explicitly called "internetworks" or "internets" (such as those based on the DARPA Internet Protocol [57] or the Xerox Internet Datagram Protocol [67]), but also some that are simply called "networks" (such as traditional store-and-forward systems like the Arpanet [51]), as well as "extended LANs" (such as those interconnected by datalink-layer bridges [39]).[3] Each of these systems has its own terminology to describe the same basic components of an internetwork; in the description of our algorithms, we use the terms shown in the figure, with the following meanings:

[3]However, the model is not intended to include systems that perform the forwarding function on higher-level units than packets, such as systems of electronic mail relays or file-transfer gateways. The term *packet* is used with its conventional meaning as the basic, addressable unit of transfer across a communications link, typically limited to a few hundred to a few thousand bytes in length.

**Subnetwork:** a communications channel or medium over which two or more devices can exchange packets. Examples are local-area networks, point-to-point serial lines, or broadcast satellite channels.

**Host:** a device that acts as a source and/or destination of packets.

**Router:** a device that forwards packets between subnetworks, so that hosts can reach destinations on different subnetworks than their own.[4]

**Subnetwork Layer:** the protocol layer that provides the service of delivering packets to and from other devices attached to the same subnetwork. This service is accessed by higher layers via the *subnetwork service interface*. A router (or a multihomed host) has a separate subnetwork layer for each type of subnetwork to which it is attached.

**Internetwork Layer:** the protocol layer that provides the service of delivering packets to and from devices attached anywhere in the internetwork. This service is accessed by higher layers via the *internetwork service interface*. The packet routing and forwarding function within a router (including multicast routing) occurs at this layer.

**Upper Layers:** the protocol layers or applications above the internetwork layer, which are the ultimate sources and sinks of packets.

Table 1.1 shows the correspondence between our terminology and that used in internetworks based on the DARPA Internet Protocol (IP), on the ISO Connectionless Network Protocol (CLNP), and on IEEE 802 Media Access (MAC) layer bridging.

It should be noted that, in a real internetwork, more than one protocol layer may fit our abstract model. In particular, any layer that performs packet forwarding may be modeled as an internetwork layer, with the potential to benefit from one of our

[4]In most systems, a router device may also act as a source and destination of packets. In our model, the term *router* refers only to the routing and forwarding functions of a device; when sourcing and sinking packets, a router device is acting in the role of a host. A device that is attached to more than one subnetwork but does not forward packets between those subnetworks is called a ***multihomed host***.

| Model Term | DARPA IP | ISO CLNP | MAC Bridging |
|---|---|---|---|
| internetwork | internet | global network | extended LAN |
| subnetwork | local network or subnet | subnetwork | LAN segment |
| host | host | end system | end station |
| router | gateway or router | intermediate system | bridge |
| subnetwork layer | link layer | subnetwork layer | physical layer |
| internetwork layer | internet layer | network layer | MAC layer |
| upper layers | transport layer and above | transport layer and above | LLC layer and above |
| packet | datagram | network protocol data unit (NPDU) | frame |

Table 1.1: Comparison of Terminology

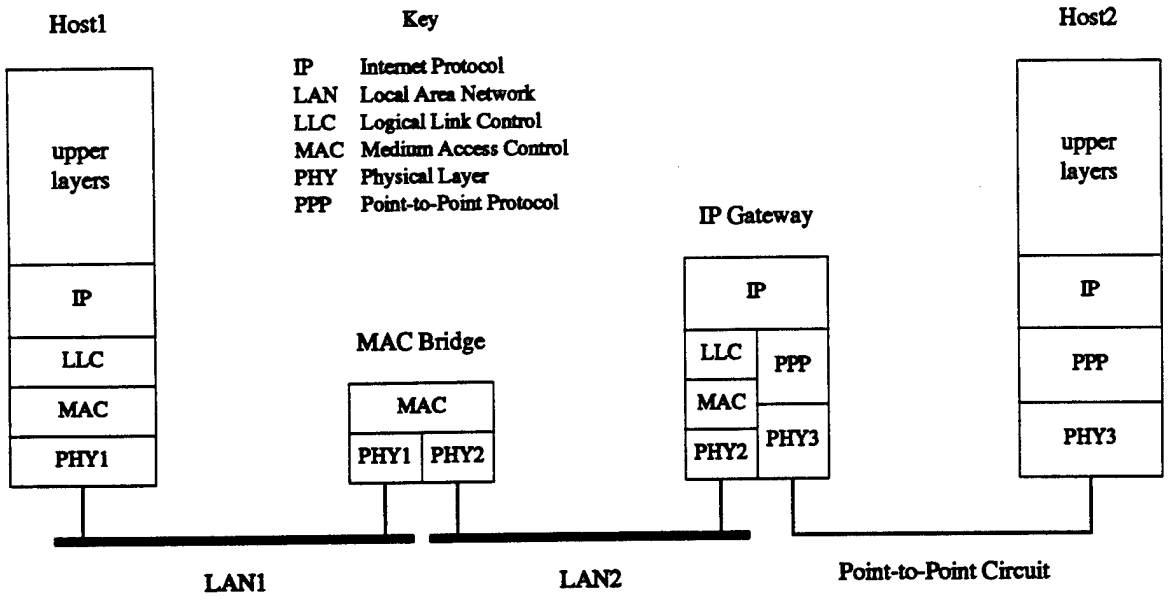



Figure 1.2: Example of Packet Forwarding at Multiple Layers

multicast routing algorithms. This can be seen in the concrete example of Figure 1.2. In this example, the two subnetworks labeled LAN1 and LAN2, and the three devices attached to them, form one internetwork, in which the MAC protocol layer plays the role of internetwork layer, the MAC Bridge plays the role of router, and Host1 and the IP Gateway play the role of hosts. Host 2 is not part of that internetwork. At the same time, there is a higher-level internetwork, containing Host1, Host2, and the IP Gateway, using IP as an internetwork layer. In this higher-level internetwork there are again two subnetworks: the point-to-point circuit between the IP Gateway and Host2, and the subnetwork to which the LLC layer offers access. The fact that the LLC subnetwork is internally composed of multiple LANs and a bridge is unknown and insignificant to the IP layer.

In the later descriptions of multicast routing algorithms, we make the following assumptions about the services offered by the (abstract) subnetwork and internetwork service interfaces:

- The subnetwork service interface offers a *datagram (connectionless) service* to the internetwork layer. That is, the internetwork layer can exchange packets with any other device on the same subnetwork without first requesting the

establishment of a connection or virtual circuit to that device. The subnetwork layer does not guarantee that packets will not be lost, damaged, duplicated, or re-ordered in transit, but such occurrences are assumed to be infrequent. If the underlying real network is a connection-oriented one, such as an X.25-based network, it is the responsibility of the subnetwork layer to hide that fact from the internetwork layer (for example, by dynamically setting up connections in response to packet transmission requests and automatically tearing down connections after an interval of no packet traffic). The internetwork layer, in turn, offers a datagram service to the upper layers.

- The subnetwork service interface does *not* require the internetwork layer to specify a *route* when sending a packet, only a destination address. If the underlying real network is one that requires the specification of a route, such as an extended LAN built from IEEE 802.5 source-routing bridges [33], it is the responsibility of the subnetwork layer to hide that fact from the internetwork layer (for example, by maintaining a table to map destination addresses into appropriate routes.) The internetwork layer, in turn, does not require upper layers to specify a route when sending a packet. (However, the internetwork may itself act as "one hop" in a higher-level packet delivery system based on source routing, implemented by a protocol layer above the internetwork layer.)

- The subnetwork service interface offers a *multicast service* to the internetwork layer. That is, it enables the internetwork layer to pass a single packet to the subnetwork service interface, for delivery to an arbitrary set of destinations on the subnetwork. We assume that, for most subnetworks, the multicast service is of the type offered by Ethernet or other LANs conforming to the IEEE 802.2 addressing scheme, in which there is a large space of addresses available for identifying multicast groups. The host interface hardware for such a LAN is assumed to include a *multicast address filter* that can recognize and discard packets destined to multicast addresses in which the host has no interest, without interrupting local processing. A router interface to such a LAN is further assumed capable of receiving packets destined to any of a *range* of multicast

addresses, without explicitly identifying every address in the range.

If the underlying real network is one that does not provide a multicast service of the type assumed, it is the responsibility of the subnetwork layer to hide that fact from the internetwork layer. This may be done in a number of ways, for example:

- if the underlying network offers only a broadcast service, rather than a multicast service, or if the host or router interfaces do not provide adequate filtering of multicast addresses, the subnetwork layer can use the underlying service to deliver packets to more destinations than required, and then filter out the unwanted packets in software at the receivers.
- if the underlying network offers only a unicast service, the subnetwork layer may emulate a multicast service by using multiple unicast packets (perhaps with the help of a centralized or distributed packet replication service that maintains lists of multicast group members). Alternatively, the services of the underlying network may be offered to the internetwork layer as a collection of virtual point-to-point links, each modeled as a separate subnetwork. (Of course, if the underlying real network is a store-and-forward network, it would be simpler and more efficient for the forwarding devices of that network to perform multicast routing and forwarding—applying our model and perhaps one of our algorithms at that lower layer—than to require the host subnetwork layer to "fake" a multicast service.)
- if the underlying network is a (real or virtual) point-to-point link, nothing special need be done in the subnetwork layer to provide multicast service, since any multicast group on that "subnetwork" can have only one member (besides the sender) and that member can be reached simply by sending a packet on the link.

The multicast service offered, in turn, by the internetwork layer to upper layers is the topic of Chapter 3.

The architectural model and assumptions we have described are satisfied by many existing packet forwarding systems. The multinational DARPA IP-based internetwork known as "The Internet" [56] is a well-known and important example. Other examples are the many private internetworks based on such proprietary protocol suites as XNS [67], DECnet [27] and AppleTalk [61], numerous collections of LANs joined by datalink-layer bridges, and emerging internetworks based on ISO CLNP.

## 1.3 Dissertation Overview

The next chapter surveys related work in the area of store-and-forward multidestination routing, and discusses its applicability to the specific problems of multicast routing in current datagram internetworks.

Chapter 3 defines a new multicast service model called the Host Group Model, with the following significant features:

- Host groups of unbounded size and dynamic membership, to which any host (not just group members) may send.
- Multicast propagation limits and bounded-scope host groups, to allow senders to reach only a "nearby" subset of a host group and to improve the scalability of multicasting in large internetworks.
- A requirement for multicast service quality (i.e., delivery performance and reliability) close to that of unicast.

The Host Group Model is suitable for a wide variety of multicast applications, is a natural extension of the unicast service offered by datagram internetworks, and is amenable to efficient implementation in large-scale internetworks.

Chapter 4 defines the multicast routing problem, in the context of datagram internetworks. The quality-of-service requirements of the Host Group Model constrain the solutions to ones in which a multicast packet is routed along the same (or an equivalent) path as a unicast packet, on its way to any one member of the destination multicast group; for internetworks that route unicast packets via shortest paths, this

means that a multicast is routed along a *shortest-path tree*. Within this constraint, we identify two variants, called *multicast delivery* and *truncated-broadcast* delivery, which offer a trade-off between the amount of routing control traffic and the amount of data traffic required to deliver multicast packets to all destination group members.

Chapter 5 describes a protocol that operates between the hosts and the routers attached to a single subnetwork, which tells the routers which host groups are present on the subnetwork. This information is required for each of our specific routing algorithms. By using a separate protocol for this purpose, hosts can be insulated from the details of any specific routing algorithm, and the routing algorithm can be changed without changing any host software.

Chapters 6, 7, and 8 present our multicast extensions to the spanning-tree, distance-vector, and link-state routing algorithms, respectively. In each case, a truncated-broadcast and a multicast version is described and analyzed. In all cases, we show that multicast routing can be performed in a single-level (non-hierarchical) internetwork with very low overhead (a few percent of the available resources), based on some reasonable assumptions about multicast traffic patterns, frequency of group membership changes, and internetwork size.

Chapter 9 describes how multiple single-level internetworks may be combined hierarchically to extend a multicast service across much larger internetworks. It also discusses the use of bounded-scope groups to further improve the scalability of the multicast service in large internetworks.

Chapter 10 covers three peripheral issues in internetwork multicast routing, that are independent of specific routing protocols. They are: (1) the behavior of multihomed hosts with respect to originating and receiving multicast packets, (2) the possibilities for exploiting multiple paths between a source and any destination group member, when delivering multicast packets, and (3) some basic limitations on the reliability and performance of multicast traffic, relative to unicast traffic.

Finally, the concluding chapter reviews the properties of the various multicast routing algorithms, and identifies some criteria for choosing among them. It also points out some outstanding issues in the area of internetwork multicasting not directly addressed in this work. It closes with a summary of our main contributions.

# Chapter 2

# Related Work

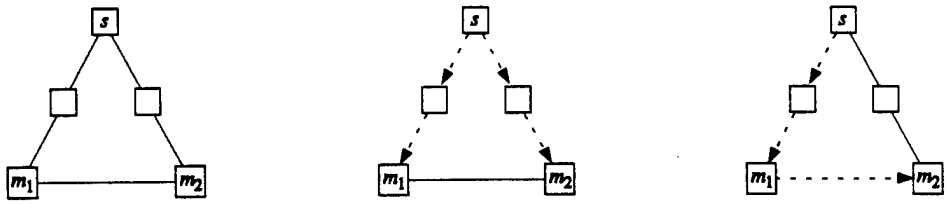
The value of a multicast service has long been recognized, and much work has been done, and continues to be done, in the design of multicast services for packet-switched networks. So far, however, the availability of multicast service has been limited mainly to those networks in which it can be provided for "free", such as multi-access LANs and broadcast satellite networks. Most of the work on store-and-forward multicasting has not yet found its way into production networks, despite the fact that one of the primary benefits of multicast—bandwidth efficiency for multiply-destined traffic—is often more significant in typically bandwidth-limited store-and-forward networks than in bandwidth-rich LANs and satellite networks. The one significant exception to this lack of store-and-forward multicast capability is found in datalink-layer bridges, which can convey broadcast and multicast packets among a set of interconnected LANs.

One reason for the limited availability of store-and-forward multicast service may be the lack of agreement on what such a service ought to look like. For *unicast* service, debates continue over what is an appropriate service model: connectionless or connection-oriented, blocking or non-blocking, reliable or "best-efforts" delivery, and so on. The work on multicast services has shown an even greater diversity in service models. For example, there is a range of possible multicast connection models, such as simplex connections from one source to set of destinations, duplex connections emerging from and converging on a single host, "omniplex" connections among a set

of hosts, or simple datagram communication without connections. Similarly, there is a wide variety of possible reliability models, of multicast group semantics, and of performance and cost goals.

In a landmark paper, Dalal and Metcalfe [24] surveyed five previously-known methods for providing multidestination delivery in a store-and-forward network based on point-to-point links, such as the Arpanet [51]. The methods are: (1) transmission of multiple unicast packets, (2) transmission of packets bearing multiple unicast addresses, (3) hot-potato forwarding, (4) spanning-tree forwarding, and (5) source-based forwarding. Those methods differ widely in such factors as their suitability for multicast rather than broadcast, their degree of reliability, their cost to the network, and the delays that they impose on delivery to all destinations. (Two of the methods—spanning-tree forwarding and source-based forwarding—underlie our multicast extensions to spanning-tree routing and link-state routing, respectively, and are described in Chapters 6 and 8.) To those five methods, Dalal and Metcalfe added two new broadcast algorithms, called *reverse-path forwarding* and *extended reverse-path forwarding*. The reverse-path algorithms are nearly as good as, or better than, any of the previously-known algorithms in terms of network overhead and delivery delay, and they have adequate reliability for a best-efforts datagram service. Two shortcomings of those algorithms, for our purposes, are that they do not work correctly in an internetwork containing multi-access subnetworks, such as LANs, and they support only broadcast delivery, not multicast. In Chapter 7, we describe those problems and present a new, multicast variant of reverse-path forwarding suitable for general internetworks that employ distance-vector unicast routing.

Another important early contribution was that of Wall [65], who emphasized the division of multidestination routing strategies into two categories: *low-delay* and *low-cost*. The difference between these two approaches can be seen in the simple topology illustrated in Figure 2.1(a). Assume that node $s$ is transmitting a multicast packet to a group consisting of nodes $m_1$ and $m_2$, and that all links have the same cost (for example, bandwidth or monetary cost) for conveying a packet and the same transmission delay. Figure 2.1(b) shows a routing in which each of the group members receives its copy of the multicast with the minimum delay of 2 hops; the total cost




(a) example topology (b) minimum delay (c) minimum cost

Figure 2.1: Minimum-Delay vs. Minimum-Cost Multicast Routing

of delivering the multicast is 4 hops. Figure 2.1(c), on the other hand, shows a routing that delivers the multicast with the minimum cost of 3 hops, but the delay to one of the members is greater than the minimum. (In graph-theoretic terms, a minimum-delay routing corresponds to a *shortest-path tree*, while a minimum-cost routing corresponds to a *minimum Steiner tree*. A minimum *spanning* tree is a minimum Steiner tree that includes all nodes in the graph, that is, a broadcast tree.) Except in special-case topologies, minimum-cost multicast routing is achieved at the expense of delay, and vice versa.

One difference between minimum-delay and minimum-cost routing is that for *closed groups*, that is, groups to which the only senders are themselves members of the group, only one tree is needed to achieve minimum-cost delivery, regardless of sender; minimum-delay delivery, on the other hand, requires multiple trees, one rooted at each sender. Wall's main contribution was a compromise between the two approaches, called *center-based fowarding*, which uses a single tree, rooted in the "center" of a group of nodes, to route multicasts among those nodes; the advantage of such a tree is that it can have low—but often not minimum—cost, while incurring low—but not minimum—average delay for multicast delivery from multiple senders. However, for *open groups*, in which the senders are not constrained to be members, both Wall's algorithm and minimum-cost routing require separate, per-sender trees, losing their advantage over minimum-delay routing in that respect. Many, perhaps most, multicast applications have a client-server or producer-consumer structure, in which the senders are not members of the multicast group, and therefore require open groups.

Another important difference between minimum-delay and minimum-cost routing

is that the problem of finding a minimum-delay routing (a shortest-path tree) has several efficient solutions, embodied in existing unicast routing algorithms, whereas finding a minimum-cost routing (a minimum Steiner tree) is an NP-complete problem. The research in minimum-cost routing, therefore, has been directed at finding good heuristic algorithms for computing low-cost multicast trees. Bharath-Kumar and Jaffe [8] examined a number of such heuristics, and compared them with minimum-delay algorithms. Their main conclusion was that the performance of the minimum-delay algorithms was "surprisingly good," in terms of network cost, compared to the average performance of their cost-minimizing algorithms: "typically only 20 percent worse" over a large number of randomly-generated topologies. More recent work in low-cost multicast routing has been based on changing some of the assumptions made in previous work. For example, Belkeir and Ahamad [5] developed low-cost multicast routing heuristics suitable for groups that have dynamically-changing membership, and McKinley and Liu [49] looked at low-cost multicasting in both regular and irregular networks of buses (such as LANs) rather than networks of point-to-point links. The importance of these low-cost schemes has diminished as the resources available in modern internetworks—bandwidth, processing, and memory—have become cheaper and more abundant; on the other hand, low delay delivery remains essential for supporting many multicast applications, such as interactive conferencing, parallel computing, and resource location.

In the specific context of internetworks, Boggs [11] introduced the notion of "directed broadcast", which is a special case of multicast in which the destination of a packet may be the set of ***all*** hosts attached to any ***one*** subnetwork in the internetwork. Directed broadcast is trivial to implement in an internetwork where each subnetwork supports broadcast, as do most LANs, and it has been designed into the XNS [67], DARPA IP [53], and AppleTalk [61] internetwork protocols. Unfortunately, it is not a particularly useful special case. It is rarely the case that the set of all hosts attached to one subnetwork corresponds to the set of hosts that a sender wishes to reach; if the desired destinations happen to be on different subnetworks, the sender must send multiple packets to reach them, and if the desired destinations are a subset of the hosts on one subnetwork, the rest of the hosts on that subnetwork suffer

the overhead of receiving unwanted packets. Because of the latter problem, Boggs suggested that directed broadcast not be used for "streaming" applications (where the efficiency benefit of multicasting is greatest), but only for logical-addressing-type applications and applications requiring a very low rate of multi-destination delivery (such as disseminating routing tables or advertising services).

Another notion introduced by Boggs is the "expanding ring search", which is a technique for locating an object or service in an internetwork. In the form described by Boggs, a searching host sends a directed broadcast query first to its attached subnetwork(s), then to subnetworks one hop away, then two hops, and so on, until it either receives an answer from the desired service, exceeds a hop limit, or runs out of subnetworks to try. This technique requires that the sender maintain or acquire a copy of the internetwork routing table, so as to know how far away each subnetwork is from itself. In the next chapter, we describe a similar mechanism that uses multicast rather than directed broadcast, and that does not require any topological knowledge on the part of the sender.

Aguilar [2] proposed an extension to DARPA IP to allow a datagram to carry multiple (unicast) destination addresses. Such a datagram would be replicated and sent in multiple directions only at those points in the internetwork where the routes to the different destinations diverged. Unfortunately, Aguilar's scheme is unable to take advantage of subnetwork multicast service, does not scale well to handling large destination groups, and requires the sender to know the identity of all of the destination group members.

Forgie [32] designed the Internet Stream Protocol to support multi-participant, real-time conferencing in the DARPA Internet. The Stream Protocol departs from the datagram model of IP, implementing a connection-oriented service that includes support for multicast connections. It is useful only for that subset of multicast applications that are satisfied with closed groups, that have the property that the senders know the identities of all the group members, and that can tolerate the delay of connection set-up before multicasting.

In the area of LAN bridging, Perlman's algorithm [55] provides for broadcast delivery via a single spanning tree computed over an interconnected set of LANs; multicast

is accomplished simply by having hosts ignore unwanted packets, using address filtering hardware in their LAN interfaces. (We have much more to say about Perlman's algorithm in Chapter 6.) Sincoskie and Cotton [62] proposed a new algorithm for multicast routing in large-scale bridged LANs, but their algorithm is limited to the support of closed groups only. The Universe Network [66] was an internetwork built by bridging together a collection of LANs with a single broadcast satellite channel; both broadcast and multicast services were easily provided, due to the broadcast nature of the satellite "backbone," but the techniques use for multicasting in that network do not generalize to internetworks of arbitrary topology.

There is clearly still a need for new internetwork multicast routing protocols that satisfy the functional and performance requirements of a wide range of possible multicast applications, and that can be implemented efficiently in large-scale, wide-area internetworks. Almost all new wide-area network services, such as SMDS [6], Frame Relay [29], and Broadband ISDN [42], are being specified with multicast as a part of the offering, though there is considerable uncertainty about what those services will look like and how they can be provided. Transport-layer multicast protocols continue to be an active area of research, for example in the work of Birman et al. [9], Cheriton [17], and Chesson [58], based on the assumption that the network and internetwork-layer multicast routing services will be available. And there is new work being undertaken to extend the Open Systems Interconnection (OSI) architecture and protocols to encompass multicast services [43]. This dissertation contributes new solutions that should be of interest in all of these activities.

# Chapter 3

# The Host Group Model

The *Host Group Model* is a new multicast service model for datagram internetworks. It defines what the multicast service looks like to users of the internetwork service interface, within a host (see Figure 1.1 on page 5); it does not define how that service is implemented. This chapter describes the Host Group Model and shows how it satisfies the functional and performance requirements of a wider variety of multicast applications than do previous models.

## 3.1 Host Groups

Under the host group model, the set of destinations of a multicast packet is called a *host group*, and it is identified by a single *group address* or *multicast address.* To accomplish a multicast, a sender simply places a group address, rather than an individual (unicast) address, in the destination address field of a packet.

As pointed out in Section 1.1, the use of group addresses allows a multicast service to be used not only for efficient multi-destination delivery, but also for *logical addressing*, that is, for reaching entities whose individual host addresses are either unknown (to the sender) or changeable—a sending host need know only a group address to reach all hosts belonging to that group.

Group addresses are drawn from the same address space as unicast addresses, so that they may be used unambiguously in a packet's destination address field. This

makes it possible, in some applications, to substitute a group address for a unicast address, without modifying the application. For example, an application that must be configured with the host address of a particular network service may instead be configured with a group (logical) address for that service so that it need not be reconfigured if the service moves to another host. On the other hand, some applications need to be able to recognize group addresses as distinct from unicast addresses. For example, a server that handles queries both as an individual and as a member of a group may respond differently to unicasts and to multicasts, perhaps suppressing negative responses to multicast queries, or randomly delaying responses to multicasts to avoid congesting the client with multiple, concurrent responses. Therefore, group addresses are encoded so as to be easily distinguished from unicast address, for instance, by using a unique prefix.

Host group addressing is directly supported by standard LANs such as Ethernet; an internetwork multicast service based on host groups is best able to exploit the efficiency and high performance of those LAN multicast services.

Other addressing schemes that have been used or proposed for internetwork multicast service are the following:

- Aguilar [2] proposed an extension to DARPA IP to allow a packet to carry more than one (unicast) destination address, by placing the additional addresses in the options field of the IP header. Compared to the Host Group Model, such a multicast scheme has a number of drawbacks: It does not support logical-addressing applications, since the sender must know the individual addresses of all destinations. It cannot effectively exploit LAN multicast capabilities. Due to the limited size of the IP options field, it artificially limits the number of hosts that can be reached by a single transmission. And the use of multiple addresses per packet degrades performance in hosts and routers, by requiring them to perform multiple route look-up operations for each packet.

- Dalal and Metcalfe [24] identified an addressing scheme, similar to Aguilar's multiple addresses, in which the destinations of a multicast packet are specified by a fixed-length *bit map*, in which each bit corresponds to a different

host. The Internet Stream Protocol [32], a connection-oriented protocol that is implemented in a few routers of the DARPA Internet, adopted the bit-map approach as a way to send a packet to a subset of the hosts in a multicast connection. Like Aguilar's scheme, the bit-map approach does not support logical addressing (the sender has to know what bits to set) and does not map easily onto LAN multicast services. The size of the bit-map either limits the total number of hosts that can be addressed or, if the bits are dynamically associated with hosts, limits the number of hosts reachable by a single transmission and requires additional protocol to handle the dynamic bit assignments. (Interestingly, the bit-map addressing feature was omitted in the latest version of the Internet Stream Protocol [64].)

- An appendix to the IEEE 802.5 Token Ring specification [38] describes a proposed multicasting scheme for interconnected rings based on bit-maps, in which the bits correspond to different *groups*, rather than different hosts. This allows a single packet to be sent to the *union* of two or more groups, which might be considered an advantage over the Host Group Model. However, this advantage comes at the cost of severely limiting the total number of groups that may exist (the number of bits in the bit-map.) Furthermore, there appear to be few applications that could benefit from this capability; applications are more likely to benefit from the ability to reach a *subset* of a group or the *intersection* of two or more groups. Section 3.2, below, describes a mechanism for reaching a subset of a group under the Host Group Model.

- A proposal for extending the OSI Reference Model to encompass "multipeer data transmission" [43] allows a packet to carry a list of destination addresses, where each address may be either an individual address or a group address. This is more general than the Host Group Model (like the IEEE 802.5 bit-map scheme, it allows transmission to a union of groups), but the extra generality does not appear to bring significant benefits. On the other hand, it introduces the complication and performance degradation of variable-length headers, and requires extra mechanisms to avoid duplicate delivery when group memberships

overlap.

Host groups have a number of additional properties that contribute to the flexibility and generality of the Host Group Model and distinguish it from some other multicast services:

**Senders need not be members.** That is, any host may send a packet to any group, whether or not that host belongs to the group. Such groups are called *open*, in contrast with *closed* groups that allow only the members of a group to send to that group. Closed groups are adequate for some "peer-to-peer" applications, such as conferencing and certain parallel computing algorithms, and several proposals have been made for supporting closed-group multicasting in internetworks, as discussed in the previous chapter. However, there are many other applications of multicast in which there is a "client-server" or "producer-consumer" relationship between the senders and the group members, for which it is inappropriate or infeasible to include all senders in the destination group. Examples are clients of a distributed database that send multicast queries to a group of database hosts, hosts that use multicast to search for or advertise a particular service, or real-time sensors that multicast status information to a group of monitoring stations.

The effect of closed groups is easily achieved under an open group model, simply by having group members discard packets received from non-members. Applications that require closed groups would be expected to operate a higher-layer protocol to control group membership, by which each member would learn the addresses or some other identification of all other members, enabling it to recognize and ignore packets from non-members. (Hosts may also be prevented from sending to particular groups by network management, using the "policy filtering" capability of most current routers and bridges.)

**Groups may have any number of members.** Multicasting schemes based on sending a packet to a list of individual addresses or to a bit-map of hosts impose an artificial limit on the number of hosts reachable by a single multicast packet (i.e., the maximum size of the list or the number of bits in the bit-map). While

those limits may be large enough for many applications, other applications can make good use of groups with a large number of members, for example, a group of recipients of a network news service or software distribution service.

If, for some applications, it is desirable to impose a limit on group size, that can be accomplished by means of a higher-layer membership control protocol.

**There are no topological restrictions on group membership.** One of the goals of an internetwork service is to make a collection of subnetworks look like and act like a single logical network, hiding the details of particular subnetworks and their connection topology. Consistent with that goal, it is preferable for the multicast service to place no constraints on the topological location of members of a group. Regrettably, the only multicast service currently available in most internetworks is a service called *directed broadcast*, which allows multicasting to a topologically-defined set of hosts, in particular, the set of *all* hosts attached to any *one* subnetwork in the internetwork. As discussed in the previous chapter, directed broadcast is a poor substitute for general multicast, and unsuitable for high-volume multicast applications, such as bulk data distribution, news dissemination, or real-time audio or video streams.

Although there are no topological constraints on group membership, the Host Group Model supports administrative constraints. In particular, it allows multicast transmissions and group memberships to be limited to a single administrative domain. This topic is discussed in Section 3.2, below.

**Membership is dynamic and autonomous.** That is, hosts may join and leave host groups at any time, with no need to synchronize or negotiate with other members of the group or with potential senders to the group. In this respect, the Host Group Model differs from most connection-oriented multicast models, in which the requirements on reliability of packet delivery require group-wide synchronization of all membership changes, and in which permission to join or leave a group must be obtained from one host (typically the originator of the connection) and must be enforced by the internetwork. Dynamic membership is important for applications, such as conferencing, where participants may come

and go independently, and for services that come and go as the hosts providing those services fail and restart. Allowing each member the autonomy to join and leave at will eliminates any requirement for the internetwork to mediate group membership, a service that not all applications require.

Particular applications may employ higher-layer protocols to control when hosts may join or leave a given group. Since the internetwork allows any host to join any group, protection against unwanted eavesdropping must be accomplished by encryption or by management constraints on routing.

**Host groups may be permanent or transient.** A permanent group has a well-known, administratively-assigned group address. It is the address, not the membership, of the group that is permanent; at any time a permanent group may have any number of members, including zero. Permanent group addresses are mainly useful as logical addresses for locating (or advertising) common services, such as bootstrap service or name service, where the specific address of the service (or the clients of the service) are unknown to the sender. Since they are permanently assigned, they may be programmed into application programs or stored in ROMs, thus eliminating the need for manual configuration.

Transient groups use temporary group addresses, and are considered to exist only for as long as they have at least one member. Multicast applications such as conferencing or distributed computing use a transient group address for the duration of a single conference or computation, after which that group address becomes eligible for reuse by another application.[1] An example of an existing network that supports transient host groups, but not permanent groups, is the DARPA Wideband Packet Satellite Network [30]; its group mechanism was

[1]The mechanism by which transient group addresses are allocated and reclaimed is independent of the service model presented here. We anticipate the use of several different techniques for allocating different portions of an internetwork's multicast address space. For example, there may be a number of servers that can be contacted to acquire a new transient group address. Some higher-layer protocols may generate higher-level transient "process group" or "entity group" identifiers which are then algorithmically mapped or hashed to a subset of the host group addresses. A range of host group addresses might even be set aside for random allocation by applications that can tolerate reception of unwanted datagrams from other multicast users; such applications could, perhaps, try several different group addresses from that range until a suitably "quiet" one is found.

designed specifically to support multi-media, multi-participant conferencing, for which transient groups are well-suited.

These properties make the Host Group Model less restrictive than other internetwork multicast models and, therefore, useful to a wider range of multicast applications.

The properties listed above are also satisfied by the multicast service available on LANs such as Ethernet, and by collections of LANs joined by datalink-layer bridges. In fact, the Host Group Model may be considered the internetwork-layer realization of the typical LAN multicast service, just as the unicast model in datagram internetworks is analogous to LAN unicast service. This commonality has significant benefits. First, the implementation of the Host Group Model is straightforward, because it maps naturally onto the subnetwork-layer service. Second, higher-layer multicast protocols and applications developed for the LAN environment may easily adapted for use in an internetwork, since the service model remains the same. Third, the LAN multicast service model is one that has already been shown to be suited to a wide range of applications—as mentioned in the previous chapter, it is the only type of multicast service that has become widely available so far.

## 3.2 Multicast Scope

One important difference between LANs and internetworks is that, while LANs—by definition—have a small geographic range and typically serve only a single community or administrative unit, internetworks may span the globe and serve a large number of organizations. Therefore, in an internetwork, unlike a LAN, it is meaningful to talk about some destinations being "closer" to a sender than others, either in terms of geographical or topological distance (e.g., number of subnetwork hops) or "administrative distance" (a host belonging to the same organization as the sender is administratively closer than a host belonging to a different organization). Some applications of internetwork multicast can benefit from the ability to limit the *scope* of a multicast transmission, to reach only "nearby" destinations.

There are a number of reasons for limiting the scope of a multicast, such as the following:

- When using multicast to locate a particular service, such as bootstrap service or printer service, the sender may not trust, or be authorized to use, servers beyond its own administrative domain.

- Some information that is multicast may be meaningful only to nearby members of a group, for example, multicast reports of unusual network events may be of interest only to nearby members of a group of network management stations.

- When sending a query to a large group, it may be preferable to reach only a few of the members, so as not to be inundated with replies, and to avoid having every member service every request. An example might be an internetwork-wide group of directory servers.

There are two general ways in which multicast scope might be incorporated into the service model. First, senders could specify the desired scope of a multicast packet by the use of an additional field in the internetwork header or, perhaps, in an existing field such as the "hop limit" or "time-to-live" field present in most internetwork headers. Second, the notion of scope could be associated with the groups themselves: there could be local groups (local to some administrative domain, for example) and global groups; senders would choose the scope of their transmissions by the choice of destination group address.

The second approach is analogous to the use of scope for identifiers in programming languages. Like a global identifier in a program, a global group address is *absolute*—it refers to the same object (i.e., host group) wherever it is used. On the other hand, a local group address, like a local identifier in a program, is *relative*—the object (host group) to which it refers depends on where it is used; the same local group address used in two distant parts of the internetwork identifies two different host groups. This distinction between absolute and relative addresses has another analog in typical file naming systems which support relative path names or absolute path names. There is also precedent for relative addressing in some internetwork protocols, such as DARPA IP or XNS, in which a special address can used to reach "all hosts on my attached subnetwork" (i.e., a local broadcast); scoped groups may be considered a generalization of that capability.

From the point of view of users of the multicast service, there are a number of advantages to using scoped groups, rather than sender-supplied scope limits:

- Scoped groups give the members of a group some control over who may send multicasts to them. For example, a group of printer servers that are intended to serve a particular administrative unit can join a group with scope local to that unit and be protected from receiving unwanted multicasts from hosts outside of that unit.

- Scoped groups enable more efficient use of the multicast address space, since they allow the same (non-global) addresses to be reused, perhaps for different purposes, in non-overlapping scopes (similar to the way local identifiers in a programming language can be reused in different procedures).

- In some applications, scoped group addresses can be transparently substituted for unicast addresses, as mentioned in Section 3.1, above. If the scope indication has to be carried as a separate packet field, it requires that sending applications be modified to provide that field, which is not required for unicast.

From the point of view of the provider of the multicast service, there is one very important advantage to scoped groups: they can greatly increase the scalability of the multicast service. As discussed later, in Chapter 9, the existence of a local group incurs no cost outside of the region of the internetwork covered by the scope of that group, since there is no need to provide routing to that group from outside the region. Thus if each region has sufficient resources to support its own local groups, the scalability of the multicast service depends only on the costs of supporting global groups and the rate at which they proliferate as the internetwork grows. The more applications that can take advantage of non-global groups, the better the multicast service will scale.

For applications to take advantage of non-global groups, the scope of those groups must match the requirements of the application. There are three units of scope that would cover the needs of many multicast applications:

- *Local-to-subnetwork.* This is the minimum distance that a multicast packet can travel—one hop. It is the desired scope for applications concerned with

internetwork topology, such as routing services or address resolution protocols. A single subnetwork also often corresponds to the smallest administrative unit within an organization, such as a single department or "work group".

- *Local-to-site*, such as a single corporate site or a university campus. Most use of multicast for discovering or advertising services would be restricted to a single site. For example, every site typically provides its own printing services, file services, bootstrap services, etc., to serve the hosts at that site. Site boundaries are also usually routing boundaries, where administrative controls on traffic into, out of, and through the site are imposed. They are often technology boundaries too, going from high-speed LANs within the site to slower metropolitan or wide-area networks.

- *Global.* For some applications, it is useful to be able to multicast to members anywhere in the internetwork. For example, a conferencing application is most useful if the participants can be located anywhere in the world. Large electronic mailing lists or network news distributions have no natural boundaries. A replicated database service may deploy servers at widely separated locations for increased availability, and use global multicasts for updates.

It seems likely that additional scope boundaries would be useful, both within a site (department, branch, division, etc.) and beyond single sites (metropolitan area, region, nation, continent). However, if there are many scope levels, applications will be more likely to need manual configuration, to indicate what scope to use in any particular instance.

It is useful to be able to determine the scope of a group address simply by examination. That enables routers that are on scope boundaries to recognize easily which multicast packets may pass over the boundary and which must be halted, without needing a table of mappings from group address to scope. It also helps to prevent mistakes when manually configuring applications. A simple way to indicate the scope of a multicast address is to designate a subfield of the address as an explicit scope identifier. A drawback of that approach is that it may be very wasteful of the address space, since it allocates an equal number of addresses to each scope level, whereas the

number of multicast addresses needed at each level is unlikely to be the same. (In particular, we expect many fewer applications to use local-to-subnetwork scope than the wider scopes.) This is of particular concern for internetworks like those based on DARPA IP, which have a relatively small address space to start with. An alternative is to simply designate a separate range of addresses for each scope level, with each range being sized for the anticipated demand at that level; determining the scope of a group address would simply require knowledge of the range boundaries. The feasibility of this approach also depends on there being a relatively small number of scope levels.

## 3.3 Expanding-Ring Searching

Expanding-ring searching is a technique introduced by Boggs [11], described in the previous chapter. It is used when a host wishes to find just one member of a host group (such as one of a set of equivalent servers, like directory servers); it is often preferable that that one member be the *nearest* one, in terms of delay or round-trip-time, so that subsequent interactions with that member have minimum response time. The technique consists of sending a multicast (or, in Boggs's case, broadcast) query packet first to those subnetworks that are directly attached to the sender, then to those one hop away, then two hops, and so on, until an answer is received or a limit on the search radius is reached.

Expanding-ring searching is an example of a use of multicast scope control, but at a much finer grain that the administrative scope levels described in the preceding section. Although we argued there that, in general, scoped groups were preferable to sender-specified scope limits, in this special case, the opposite is true. An expanding ring search can be performed trivially and efficiently by having the sender of a multicast packet specify how far the packet may go, using the "hop limit" or "time-to-live" field that is present in most internetwork protocol headers. The host simply starts with hop limit of 1, and increments it on each retransmission, keeping the destination address (which identifies the sought group) constant. Unlike Boggs's version, this method of expanding-ring searching does not require any knowledge of the topology

on the part of the sender.

We note that this technique provides only an approximation of the desired functionality, in that:

- the group members that are nearest in terms of subnetwork hops are not necessarily nearest in terms of delay (although that is usually the case in LAN-based internetworks).
- the multicast may be delivered to more than one group member, when they are equally close to the sender (although this presents no particular problems—the searcher simply chooses the one that responds first).
- the response from the nearest member may be lost or damaged in the internetwork, causing the searcher to choose a more distant member (although those members nearer the searcher have more opportunities to respond, since they fall within the scope of later retransmissions[2]).

The use of hop limits for fine-grain scope control is complementary to the use of scoped groups for bounding multicast traffic to administrative boundaries. For example, an expanding ring search would normally be performed using a group address with a particular administrative scope, for example, when looking for the nearest directory server within the sender's own site; hop limits provide fine control over the radius of the search, while the group address limits the total scope of the search (the maximum radius). The Host Group Model makes both hop control and scoped groups available to multicast applications.

## 3.4 Multicast Reliability and Performance

A distinguishing characteristic of a datagram service is that it offers no guarantee that packets will not be lost, damaged, duplicated, or delivered out of order; applications that are intolerant of such effects employ host-to-host transport or higher-layer protocols to compensate as required. Datagram services also do not offer any performance

[2]In that respect, our scheme differs from Boggs's, which sends only one query packet to each subnetwork.

guarantees, such as bounds on maximum delay or minimum throughput; transport protocols cannot compensate for poor datagram performance, but they may be able to specify to the internetwork layer a preferred *quality-of-service*[3] (QOS), for example, to select a low-delay delivery path over a high-throughput path. These characteristics apply equally to a multicast datagram service as to a unicast datagram service.

From the application's point of view, a datagram multicast service is most valuable if multicast packets are delivered with reliability and performance close to that of unicast packets. That is, each member of a host group should receive multicast packets sent to that group with close to the same delivery probability, delay, and throughput as unicast packets sent to the individual member (assuming, of course, that the member is within the hop limit of the multicasts). This property has a number of important benefits:

- It allows for higher-layer, reliable multicast protocols based on the same assumption as reliable unicast protocols, that is, that a small number of retransmissions is sufficient to guarantee delivery to every group member, except those members that have failed or become partitioned from the sender. (Of course, the probability of a member failing or becoming partitioned increases with the size of the group.)

- It allows the round-trip times observed during unicast exchanges with members of a group to be used for computing retransmission timers for subsequent multicast exchanges with that group.

- It ensures that there are no disincentives to using the multicast service when an application requires multi-destination delivery. This is advantageous insofar as it conserves internetwork resources to send a multicast packet rather than a sequence of unicast packets.

Much of the previous work in internetwork or store-and-forward multicasting has been concerned with minimizing the cost to the network of providing the service, at the expense of host-to-host performance and reliability. This trade-off was discussed in the previous chapter.

[3] ***Quality-of-service*** is the ISO term; in DARPA IP, it is called ***type-of-service***.

Multicasting raises some additional reliability and performance issues that do not apply to unicast service:

**Simultaneity of delivery.** One property of LAN multicasting is the near simultaneous delivery of a multicast packet to all group members, due to the physical proximity of the hosts on the LAN. Unfortunately, to retain this property in an internetwork multicast service would require that delivery to group members nearer the sender be artificially delayed until the furthest members have been reached. For any applications that require simultaneous delivery, higher-layer protocols in conjunction with synchronized clocks can provide the desired effect—packets can be timestamped with a desired delivery time by the sender, and delayed in each receiver until that time arrives.

**Misdelivery.** The dynamic nature of host group membership, and the use of broadcast technology for multicast delivery within subnetworks, raises the hazard of delivering multicast packets to non-member hosts. It is preferable to avoid misdelivery, in order not to consume resources in unintended receivers; however, it would be undesirable to give up the flexibility of dynamic groups or the efficiency of broadcast-based delivery in order to guarantee that misdelivery never occurs. The use of *multicast address filters* in subnetwork interface hardware (as discussed in Section 1.2) allows hosts to defend themselves from many unwanted multicasts. Unfortunately, the address filters in many current LAN interfaces are rather poor, either having too few address slots or too few address hash buckets to prevent the reception of unwanted multicast addresses; we expect that situation to improve as multicast services and multicast applications become more widespread. There are also cases in which more than one application may end up sharing the same multicast address, due to shortage of multicast address space (at either the datalink layer or the internetwork layer), or due to "sloppy" address allocation algorithms. In those cases, hosts have no choice but to filter out unwanted packets in software.

**Join latency.** The delay between the time a host joins a host group and the time it is able to receive packets addressed to that group is called the *join latency.*

On a LAN, the join latency is usually just the delay in updating the multicast address filter in the subnetwork interface. In an internetwork, join latency may be greater, since routers may have to be informed of the membership, as discussed in later chapters. Low join latency is important to some applications, such as those that use multicast to communicate with migrating processes or mobile hosts.

*Leave latency*, that is, the delay between the time a host leaves a group and the time it stops receiving multicasts for that group, is less of a concern, since a host can just ignore packets that arrive after departing from the group, using hardware or software address filtering.

The Host Group Model is consistent with the datagram or connectionless approach to internetworking, in that it offers access to the basic underlying transmission service with a minimum of embellishments or constraints, leaving it to higher-layer, end-to-end protocols to enhance the basic service for those applications that need it. Two examples of higher-layer protocols that offer a "value-added" multicast service across a single LAN are VMTP [18] and ISIS [10]. They each offer demultiplexing services, various degrees of control over (process) group membership, and a range of multicast delivery guarantees. The close correspondence between the Host Group Model and the service model supported by LANs facilitates the use of these and other multicast protocols and applications in an internetwork environment.

## 3.5 The Internetwork Service Interface

Under the Host Group Model, the service interface for sending and receiving internetwork packets remains the same. Typically, there are two operations:

*Send* (*source, destination, quality-of-service, hop-limit, data*)
*Receive* (*source, destination, quality-of-service, data*)

The *Send* operation requests transmission of a packet as specified by the arguments of the operation. For multicast packets, the *destination* argument is a group address.

In some internetwork protocols, a *time-to-live* argument serves as the hop limit. The *Receive* operation is invoked on the arrival of a packet, with the indicated arguments being passed to the higher layer.

In addition, two new operations are provided at the service interface:

*JoinGroup* (*group, subnetwork*)
*LeaveGroup* (*group, subnetwork*)

where *group* is the address of a host group that the the caller wishes the host to join or leave. The *subnetwork* argument is needed only if the host is multihomed, that is, attached to more than one subnetwork; it allows the caller to specify which subnetwork to listen on, for multicasts to the given group. Normally, a host would listen on one subnetwork only, to avoid reception of duplicate multicast packets. However, there are cases in which listening on multiple subnetworks may be desired, for example, when offering the same service via different subnetworks to different communities of hosts, each of which is limiting the scope of their multicasts, as discussed in section 3.2. Chapter 10 addresses this and other issues regarding multihomed hosts.

A concrete example of the Host Group Model service interface can be found in our specification of multicast extensions for DARPA IP [25].

## 3.6 Chapter Summary

In this chapter, we have described a new multicast service model for datagram internetworks called the Host Group Model. It is based on two simple extensions to the service supported by existing internetwork datagram protocols:

- Part of the space of destination addresses is reserved for identifying *groups* of hosts, rather than individual hosts.
- A pair of new operations are added to the internetwork service interface, to allow a host to join and leave host groups dynamically.

The Host Group Model offers a more general (less restrictive) internetwork multicast service than others that have been implemented or proposed. For example, unlike multicast schemes based on lists of individual addresses or bit-maps, host group

addressing supports not only multi-destination delivery, but also logical-addressing applications, and it imposes no artificial limit on the number of destinations reachable by a single multicast packet. Unlike the directed broadcast scheme supported in many current internetworks, the Host Group Model imposes no topological constraints on group membership, and unlike proposed close group models, it allows any host to send multicasts to any group. This greater generality makes the Host Group Model suitable for a wider range of multicast applications.

Consistent with the datagram approach to networking, it is left to higher-layer protocols to impose whatever restrictions on group membership and "openness" as may be required by particular applications. Higher-layer protocols are also responsible for enhancing the reliability of multicast delivery as needed, beyond the "best-efforts" datagram delivery service provided by the internetwork. Multicast packets are delivered to the members of a host group with the same delay and throughput as unicast packets to the same hosts, in order to simplify the design of the higher-layer protocols. The quality-of-service mechanisms provided by internetwork protocols may be used to influence the choice of multicast delivery characteristics provided by the internetwork service.

Support for multicast *scope control* offers further capabilities, in particular, the ability to limit the scope of a host group to a specified administrative domain, and the ability to perform an expanding-ring search to locate the nearest member(s) of a host group.

As later chapters show, the Host Group Model can be efficiently implemented in large internetworks. Of particular importance to the efficiency of the service is its close correspondence to the type of multicast service offered by modern LANs.

Another way to think of the Host Group Model is as the natural generalization of existing datagram services, in which unicasting is simply the special case of multicasting to a host group with one permanent member. However, unicasting is a sufficiently important and dominant special case that it is reasonable to optimize the delivery service for unicasting. Therefore, the mechanisms described in the following chapters for supporting the Host Group Model are not expected to replace existing unicast delivery services (although they are expected to support groups that, at any

particular time, may have only one member), and the distinction between unicasting and multicasting is maintained throughout.

# Chapter 4

# The Multicast Routing Problem

The multicast routing problem for datagram internetworks can be visualized as illustrated in Figure 4.1. A source host (the circle labeled $s$) transmits a multicast packet, destined to a host group whose members (labeled $m$) are distributed across one or more subnetworks. We assume that the subnetwork to which the source host is attached (represented by a thick, horizontal line) is capable of delivering a copy of the packet to any group members attached to that subnetwork, plus any routers (represented by boxes) attached to that same subnetwork. The multicast routing problem is how to arrange for the routers to deliver one copy of the packet to every other subnetwork to which group members are attached, across an arbitrary topology of routers and subnetworks (represented by the "cloud"). Each destination subnetwork, like the source subnetwork, is assumed capable of completing delivery to its attached members.[1]

Multicast routing is a generalization of unicast routing, and a multicast routing algorithm must deal with many of the same issues as a unicast algorithm, such as learning the internetwork topology, detecting changes in the topology, and computing delivery paths on the topology. Therefore, we have started with existing unicast

[1]Recall from Section 1.2 that, for any subnetwork that does not naturally support multicast, it is the responsibility of the subnetwork layer protocols to hide that fact from the internetwork layer in the hosts and routers. For example, in a unicast-only subnetwork, the subnetwork-layer protocols may have to simulate a multicast service, using replicated unicasts.

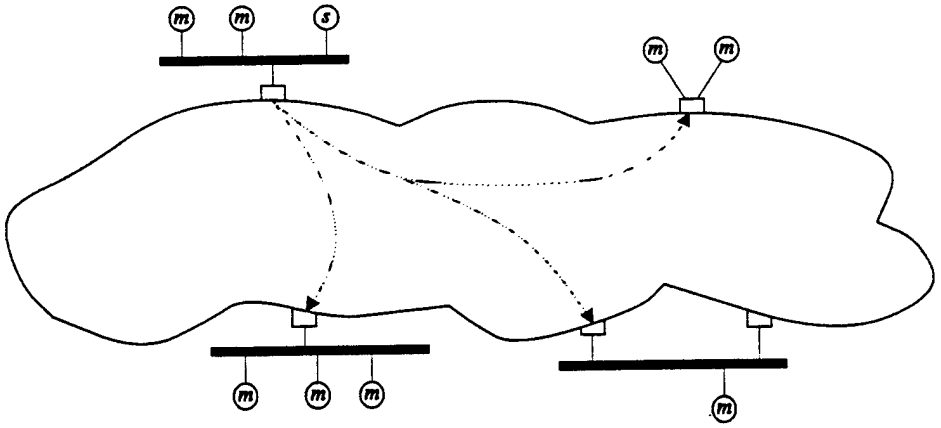



Figure 4.1: The Internetwork Multicast Routing Problem

routing algorithms and extended them to handle the more general multicast routing problem. There are a number of different unicast routing algorithms currently employed in datagram internetworks, each suited to different circumstances such as the size of the internetwork, the protocol layer at which routing is performed, or the relative scarcity of various resources (processing, memory, bandwidth, administrative personnel, etc.). We have developed multicast routing extensions to three of those: the *spanning-tree* algorithm used by most datalink-layer bridges, and the *distance-vector* and *link-state* algorithms employed by most network and internetwork-layer routers.

Under all three extended algorithms, the set of subnetworks and routers through which a particular multicast packet is forwarded forms a *delivery tree*, rooted at the source of the packet; copies of the multicast packet are generated only at those points where the tree branches. The delivery tree has the property that each destination member receives the multicast packet over the same path (or as good a path) as that over which it receives unicast packets from the same source. (In the normal case where unicast routing is done via the shortest path, this is known as a *shortest-path tree*, and if path length is measured in terms of delay, it is also called a *minimum-delay tree*.) This property ensures that multicast packets are delivered to each destination member with delay, throughput, and reliability close to that of unicast packets, as required by the Host Group Model.

A delivery tree that reaches *all* subnetworks in the internetwork is called a *broadcast tree.* A *multicast tree* may be thought of as a broadcast tree that has been pruned back so that it does not extend beyond those subnetworks that have members of the destination group. An intermediate type of delivery tree is the *truncated-broadcast tree*, which is a broadcast tree that has been trimmed of all *leaf* subnetworks except those that have destination group members. All three types of trees—multicast, broadcast, and truncated-broadcast—may be used to deliver multicast packets; in the case of broadcast or truncated-broadcast, the packets may simply go further than necessary. This is illustrated in Figure 4.2, which shows an example topology and the three types of delivery trees, for a multicast packet originating from a host on the subnetwork labeled $s$, destined to a group with members on subnetworks $m_1$, $m_2$, and $m_3$.

Two details of Figure 4.2 need further explanation:

- In the multicast case, (b), the router below $m_1$ receives a copy of the packet unnecessarily (indicated by the arrow marked "*"). This occurs because subnetwork $m_1$ delivers the multicast packet to all attached member hosts *plus* all attached routers. In this case, the packet is simply ignored by the downstream router. A similar thing happens on the subnetwork to the right of $m_1$: the packet is delivered to the (empty) set of attached member hosts plus the one downstream router. However, in this case, the router does not ignore the packet, but forwards it on to other downstream member subnetworks. This approach of including the routers as receivers of all subnetwork multicasts has the advantage that it takes only one transmission to reach all attached member hosts (if any) plus all attached downstream routers (if any), thus minimizing the processing costs in the sender, the bandwidth costs on the subnetwork and the delivery delay to all downstream subnetworks.

- In the broadcast case, (c), the transmission on the horizontal point-to-point link (indicated by the arrow marked "*") occurs because that link is itself treated as a subnetwork, which therefore receives a copy of the packet. The receiving router simply ignores that copy of the packet, because it does not arrive from

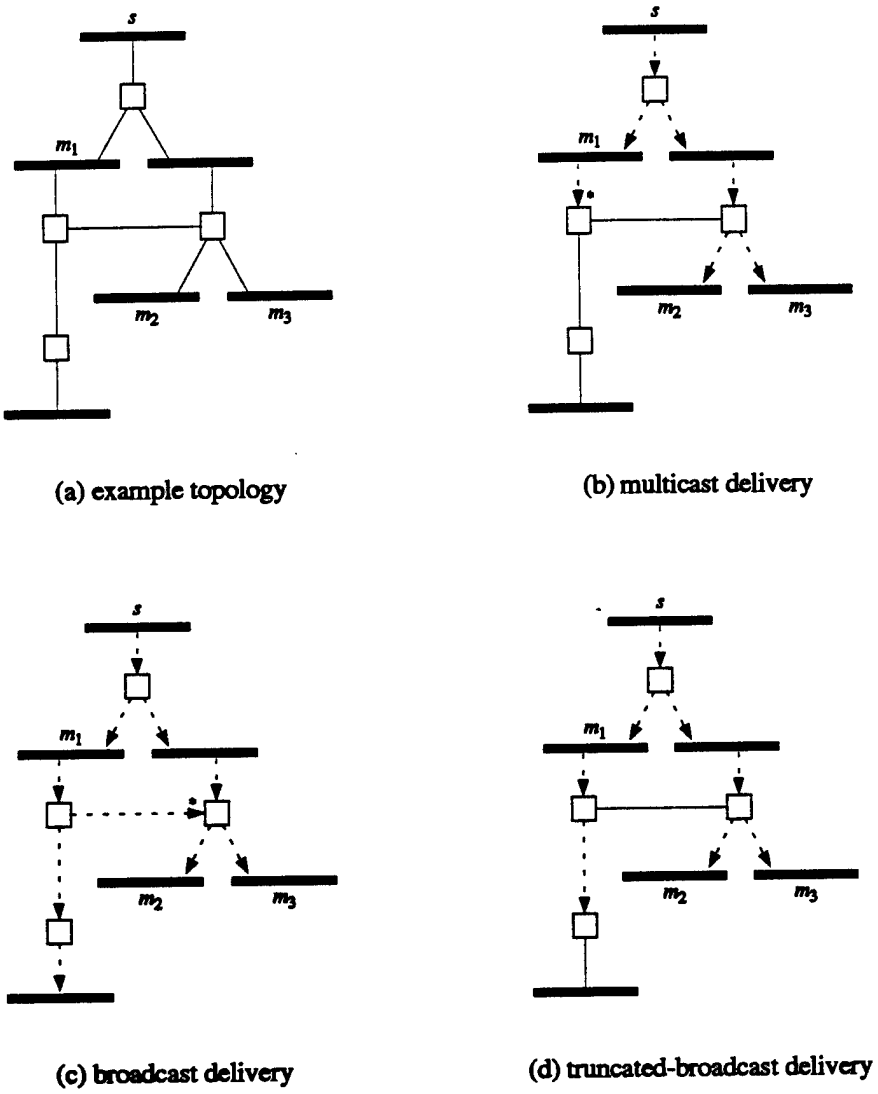



Figure 4.2: Three Types of Delivery Trees

the shortest path from the source.

In Chapters 6, 7, and 8, we describe both multicast and truncated-broadcast versions of each of the three basic routing algorithms: spanning-tree, distance-vector, and link-state. (Truncated-broadcast routing generates significantly fewer packet copies than full broadcast routing in most topologies, and is negligibly more expensive to perform, so we do not explicitly describe broadcast versions of the algorithms, except as needed to explain truncated-broadcast.) The multicast versions have the advantage of incurring no unnecessary "packet hops" when delivering a multicast packet—copies of the packet traverse only those subnetworks that either have members or are on the shortest path to members of the destination group. However, the multicast versions require that routers exchange and maintain information about where group members are located. In the truncated-broadcast versions, the need for such control traffic is much reduced—routers need know only about the members present on their directly-attached subnetworks. Therefore, the two approaches offer a trade-off between the costs of exchanging and maintaining group information and the costs of excess multicast data traffic. This trade-off is sensitive to the topology of a particular internetwork, to the nature of specific groups (for example, how sparsely distributed the group members are), and to the nature of the multicast traffic itself (for example, how frequently multicast packets are sent, and how often they are sent with small hop limits). We examine this trade-off in Chapter 11, after identifying the specific costs of each of our algorithms.

The basic algorithms are designed to operate in a single-level or "flat" internetwork, in which all routers run the same routing algorithm and maintain the same level of detail about the internetwork's topology. Due to scaling limits in the (unicast) routing algorithms, such as routing table size or bandwidth required for routing updates, those algorithms are typically limited to handling internetworks no larger than a few hundred subnetworks. To grow beyond that size, internetworks are structured *hierarchically*, by dividing the internetwork into regions (and, if necessary, sub-regions and sub-sub-regions and so on), such that each region remains small enough to be handled by one of the basic routing algorithms. At higher levels in the hierarchy, routing is done by treating the regions as if they were single subnetworks, and again applying

one of the basic algorithms to route between regions. In Chapter 9, we describe how our multicast routing algorithms can operate in a hierarchically structured internetwork, to greatly extend the range of multicast service. Different multicast routing algorithms, and different versions (i.e., multicast or truncated-broadcast) of the algorithms, may be used in different regions of the internetwork, and at different levels in the hierarchy.

All of our algorithms require some way for routers to learn which host groups are present on their attached subnetworks. Therefore, before presenting the details of the specific algorithms, we first describe in Chapter 5 a simple protocol by which hosts can report their group memberships to their neighboring routers. It is then up to those routers to distribute the information further, as required by the specific routing algorithm. (For the truncated-broadcast versions, no further distribution is needed.) The protocol keeps the hosts unaware of the specific routing algorithm in use, and insulates them from changes to that algorithm.

# Chapter 5

# The Host Membership Protocol

In this chapter, we describe a protocol by which routers can learn which host groups are present on their directly-attached subnetworks. The information provided by this protocol, which we call the *Host Membership Protocol*, is needed by the routers for both multicast delivery and truncated-broadcast delivery, and for all of the specific routing algorithms described in subsequent chapters; it is the only information required of hosts in support of multicast routing.

The protocol enables every router attached to a subnetwork to:

- learn which host groups are present on the subnetwork (i.e., have at least one member host present), at the time the router starts up,
- detect when a new host group appears on the subnetwork (i.e., the first host joins the group), and
- detect when a host group disappears from the subnetwork (i.e., no hosts remain in the group).

Since the routers' task is to deliver multicast packets to subnetworks, not to individual hosts, they need not know the identity of every host member of a group; they need only know that at least one member is present on a subnetwork.


(a) host membership query (b) host membership report

Figure 5.1: The Host Membership Protocol

## 5.1 Description of the Host Membership Protocol

The Host Membership Protocol is based on a simple query-response scheme. On each subnetwork, one of the attached routers is elected as the *interrogator*. (How the election is performed is a private matter between the routers; in later chapters, we specify election procedures for each of our specific routing algorithms.) The interrogator periodically multicasts a *host membership query* packet to the subnetwork; the query packet is sent to the *all-hosts* group, whose scope is local to the subnetwork, and to which all hosts belong. This is illustrated in Figure 5.1(a), which shows two routers attached to a subnetwork, one of whom (the interrogator, labeled $i$) is multicasting a query to all of the hosts attached to the subnetwork. (The routers' attachments to other subnetworks are obscured behind the "cloud".)

For each group that is present on the subnetwork (and that has scope wider than the subnetwork), one member host responds to the query by multicasting a *host membership report* packet. The report packet is addressed *to* the group that is being reported, and is sent with a maximum hop-limit of one. This results in delivery to all other members of that group on the same subnetwork, plus all attached routers (since the routers automatically receive multicast packets sent to any group whose scope is wider than the subnetwork), as shown in Figure 5.1(b). The report serves two purposes: it tells the routers that the group is present, and it tells the other members *not* to report the group. The routers conclude that a group is no longer present when no report is received for the group, after a small number of queries.

The detailed host algorithm is specified by the pseudo-code in Figure 5.2. This

```
1: on upper-layer request to join group g on subnetwork k:
       create new membership m; m.group ← g; m.subnet ← k
       if nonlocal(g)
           send report to g on k; m.timer ← T_N; m.count ← 2

2: on upper-layer request to leave group g on subnetwork k:
       if ∃ membership m such that m.group = g and m.subnet = k
           delete m

3: on receipt of query from subnetwork k:
       for each membership m
           if nonlocal(m.group) and m.subnet = k and m.timer = 0
               m.timer ← random(T_D)

4: on expiration of m.timer for membership m:
       send report to m.group on m.subnet
       if m.count < N
           m.timer ← T_N; m.count ← m.count + 1

5: on receipt of report for group g from subnetwork k:
       if ∃ membership m such that m.group = g and m.subnet = k
           m.timer ← 0; m.count ← N
```

Figure 5.2: The Host Membership Protocol—Host Algorithm

algorithm (and all subsequent ones) takes the form of a state machine. Each event to which the machine responds is labeled with a number for ease of reference in the description, below.

The host algorithm operates on abstract objects called "memberships"; a membership $m$ has the following attributes:

| | |
|---|---|
| $m$.group | host group address |
| $m$.subnet | attached subnetwork identifier |
| $m$.timer | retransmission/delay timer |
| $m$.count | retransmission counter |

Timers count down to 0; the event of reaching 0 is called an *expiration.* A timer can be stopped by explicitly setting it to 0, which does not count as an expiration. The predicate nonlocal($g$) is true if group $g$ has scope wider than a single subnetwork. The procedure random($x$) generates a new random value between 0 and $x$.

Whenever an upper-layer protocol invokes the *JoinGroup* operation (event 1), a new membership is created. If the newly-joined group has scope wider than a single subnetwork, the host immediately sends an unsolicited host membership report for that group, rather than waiting for the next periodic query. This informs the routers of the group's presence, in case there were no previous members present on the subnetwork—the sooner the routers learn of the group, the sooner they are able to forward multicast packets destined to the group onto the subnetwork, thus minimizing join latency. Also, the membership's timer is set to a small value, $T_N$, to trigger a retransmission of the report after a short interval, in case the initial report is lost or damaged in the subnetwork before being delivered to all attached routers. The report is transmitted enough times, $N$, to reduce the probability of delivery failure to an acceptably small value. (In the pseudo-code, the retransmission counter is initialized to 2 to account for the initial transmission plus the one that occurs when the timer expires.)

When an upper-layer protocol invokes the *LeaveGroup* operation (event 2), the corresponding membership is deleted.

When a host receives a host membership query (event 3), it does not respond immediately. Instead, for each group to which it belongs on the subnetwork and

whose scope is wider than the subnetwork, it sets its membership's timer to a random delay value between zero and some maximum delay, $T_D$; a separate random delay is chosen for each group.

When a membership's timer expires (event 4), a corresponding host membership report is transmitted. If the number of initial transmissions, $N$, has not yet been reached, the timer is reset to $T_N$ to trigger the next transmission; otherwise, the timer remains halted.

Whenever a host membership report is received (event 5), if the host belongs to the reported group, it stops its timer for that group (if it happens to be running) and abstains from sending its own report. Thus, in the normal case, only one member of each group present will generate a report in response to a query, the one whose timer for that group expires first. The single report accomplishes both the cancellation of the reports from other members and the notification of the routers that the group is present. If, on occasion, more than one member of the same group sends a report for the group (because their timers expire at sufficiently close to the same time), no harm is done, except for the consumption of an extra packet's worth of bandwidth and processing by the routers and member hosts.

Figure 5.3 shows the router algorithm. The router algorithm operates on two types of object. A "group presence" object $p$ has the following attributes:

| | |
|---|---|
| $p$.group | host group address |
| $p$.subnet | attached subnetwork identifier |
| $p$.time-left | time left until group is assumed absent |

An "attached subnetwork" object $k$ has the following attributes:

| | |
|---|---|
| $k$.interrogator | true if this router is interrogator for $k$ |
| $k$.timer | retransmission/query timer |
| $k$.count | retransmission counter |

The $k$.interrogator Boolean is set by an interrogator election algorithm which is specific to a particular routing algorithm; Figure 5.3 shows only the actions taken in response to being elected ($k$.interrogator changing from false to true) or resigning ($k$.interrogator changing from true to false)

```
1: on start-up:
       for each attached subnetwork k
              send query to all-hosts on k; k.timer ← T_N; k.count ← 2

2: on being elected interrogator for subnetwork k:
       if k.count = N
              send query to all-hosts on k; k.timer ← T_Q

3: on resigning as interrogator for subnetwork k:
       if k.count = N
              k.timer ← 0

4: on expiration of k.timer for subnetwork k:
       send query to all-hosts on k
       if k.count < N
              k.timer ← T_N; k.count ← k.count + 1
       else if k.interrogator
              k.timer ← T_Q

5: on receipt of report for group g from subnetwork k:
       if ∃ group presence p such that p.group = g and p.subnet = k
              p.time-left ← T_P
       else
              create new group presence p
              p.group ← g; p.subnet ← k; p.time-left ← T_P

6: on expiration of p.time-left for group presence p:
       delete p
```

Figure 5.3: The Host Membership Protocol—Router Algorithm

When a router starts up, it immediately sends $N$ queries at intervals of $T_N$, so that it may quickly learn what groups are present without waiting for the interrogator (if one exists) to issue the next periodic query. After the initial $N$ transmissions, if the router has been elected (or is subsequently elected) interrogator, it proceeds to send periodic queries at intervals of $T_Q$, until it fails or resigns as interrogator. This behavior is accomplished as shown in the pseudo-code for events 1–4.

Whenever a router receives a host membership report from a particular group on a subnetwork (event 5), it resets the time-left for that group to a value $T_P$. (If the router has no record of the group's presence, a new presence object is created to hold the group's timer.) If a group's time-left expires (event 6), the router concludes that the group is no longer present on the subnetwork, and deletes its record. To allow for the possible loss of individual query or report packets, as well as the random reporting delays, the presence timeout $T_P$ is chosen to be $N$ times the query interval $T_Q$, plus the maximum random delay $T_D$.

## 5.2 Costs of the Host Membership Protocol

The costs of the Host Membership Protocol to the hosts, routers, and subnetworks depend greatly on the choice of query interval $T_Q$: the larger $T_Q$, the less the overhead of the protocol. On the other hand, the larger $T_Q$, the larger the presence timeout $T_P$ must be, which is the maximum time it may take the routers to detect that a group has disappeared from a subnetwork; during that time they may unnecessarily forward multicast traffic for the departed group onto the subnetwork. A relatively large presence timeout on the order of several minutes is reasonable, for the following reasons:

- The hosts on a subnetwork are protected from receiving unwanted multicast packets, for example by the use of multicast address filters in their subnetwork adapters. Therefore, it doesn't matter to the hosts how long the routers continue to forward unnecessary multicasts onto the subnetwork (except insofar as the unnecessary multicast packets consume bandwidth that would otherwise be available to the hosts).

- Most group memberships are expected to be long-lived. For example, server applications that join a group so as to be locatable by multicast tend to join when they start up, and remain members until they fail—anywhere from hours to months. Uses of multicast that involve human participation, such as conferencing, require group memberships lasting from a few minutes to several hours. Assuming that the cost of the multicast traffic to the member subnetworks of such groups is tolerable during their lifetimes, presumably a couple of minutes more of that traffic after they are gone (assuming the traffic continues) is also tolerable.

- Many short-lived groups have the property that the multicast traffic sent to them stops when the members leave the group. Two examples are: (1) a group formed for the purpose of performing a distributed computation, in which all the members leave the group when the computation is completed, and (2) a group formed to receive a bulk data transfer, which disbands upon receiving an end-of-transfer message. In those cases, it is harmless for the routers to continue to believe that the members exist after they have departed, because there will be no further multicast traffic for them.

The other parameters of the Host Membership Protocol are:

- $N$, the number of transmissions of a packet required to make the probability of delivery failure negligible. An appropriate value depends on the specific subnetwork; modern LANs and fiber-optic circuits have very low loss rates, such that an $N$ of 2 or 3 ought to be adequate. An $N$ that is too small causes groups to spuriously "disappear" then "reappear" on the subnetwork, from the routers' point of view; such events can be monitored by network management, and the value of $N$ can be adjusted as necessary.

- $T_N$, the short interval between the initial $N$ transmissions of a host membership query (when a router starts up) and a host membership report (when a host first joins a group). This need only be long enough to avoid multiple transmissions being lost due to the same error or congestion "event" (i.e., the longer

| *Parameter* | | *Assumed Value* |
|---|---|---|
| # transmissions to assure delivery | $N$ | 2 |
| initial retransmission interval | $T_N$ | 1 second |
| maximum report delay | $T_D$ | 10 seconds |
| query interval | $T_Q$ | 50 seconds |
| presence timeout | $T_P$ | 110 seconds ($NT_Q + T_D$) |

Table 5.1: Parameters of the Host Membership Protocol

the interval, the more likely that the loss probabilities of the packets will be independent). Like $N$, it is a function of the specific subnetwork, and should be tunable by network management.

- $T_D$, the maximum random delay before a host responds to a membership query. The number of possible random values is limited by the granularity of the host clock; making $T_D$ larger increases the number of values and, therefore, decreases the probability of two members of a group choosing the same delay. With typical clock granularities of 10 msec or better, a $T_D$ of 10 seconds gives a range of at least 1000 values to choose from. (Recall, also, that the consequences of two members occasionally choosing the same delay are negligible.)

Table 5.1 summarizes the protocol parameters, and lists an assumed value for each, for the purpose of estimating the protocol costs below.

The dominant computational cost for a host that runs the protocol is the cost of receiving query and report packets, and sending report packets; we call these "packet events". For each subnetwork to which a host is attached, the host will incur $(G_h + NJ_h + 1)/T_Q$ packet events per second, where:

- $G_h$ is the number of host groups to which the host belongs on the subnetwork. For most hosts, we would expect this number to be somewhere in the range 5–20, based on observations of current LAN-based multicast applications.

- $J_h$ is the expected number of times that either this host joins a new group or some other host joins a group of which this host is already a member, in any interval $T_Q$. The distribution of such "join events" will be very non-uniform,

but the long term average should be quite small, probably no more than one per query interval.

- The "+ 1" term accounts for the reception of the periodic query packet. (We ignore the reception of queries due to router start-up, since such events should occur very infrequently.)

Using these estimates, we get a host cost of approximately 0.15–0.5 packet events per second, for each attached subnetwork.

The host storage cost depends on the choice of data structure for the host's membership information. Since we expect hosts to be members of a relatively small number of groups, a simple and adequate data structure would be a linked list for each attached subnetwork, containing records of the form:

| group | timer | count | link |
|---|---|---|---|

With a 4 byte DARPA IP group address, 2 bytes each of timer and count, and 4 bytes for linking the record into its list, the record length is 12 bytes and the estimated host storage cost is $12 \times G_h$, that is 60–240 bytes, for each attached subnetwork.[1]

For the routers, the dominant computational cost is that of handling $(G_k + NJ_k + 1)/T_Q$ packet events per second on each attached subnetwork, where:

- $G_k$ is the number of host groups present on the subnetwork. A reasonable estimate would be 10–50 groups for most subnetworks.

- $J_k$ is the expected number of times that any host joins a group on the subnetwork, in any interval $T_Q$. As above, we estimate this to be no more than one per query interval, on average.

- The "+ 1" term accounts for the transmission of the periodic query packet.

[1]The most performance-critical operation that must be supported by the host membership data structure is verifying the host's membership in the destination group of each incoming multicast packet (an operation that is not part of the Host Membership Protocol). If the number of group memberships or the rate of multicast reception warrants, more time-efficient structures than linked lists, such as hash tables or binary trees, may be used, with a corresponding increase in storage cost. In any case, the storage requirements will be small and linear in either the number of groups to which the host belongs or the number of subnetworks to which the host is attached.

Using these estimates, we get a router cost of approximately 0.25–1.0 packet events per second, for each attached subnetwork. To efficiently support the various multicast routing algorithms described in later chapters, it is best for a router to store group presence information in such a way that, given a group address from an incoming multicast packet, it can quickly determine on which of the attached subnetworks members of that group are present. A suitable data structure might be a hash table or a search tree, keyed by group address, and containing records of the form:

| group | time-left$_1$ | ... | time-left$_n$ | link |
|---|---|---|---|---|

where $n$ is the number of attached subnetworks. (A time-left$_k$ value of zero would indicate that the group is not present on subnetwork $k$.) Assuming 4 bytes of DARPA IP group address, two attached subnetworks (i.e., $n = 2$) with 2 bytes of time-left each, and 4 bytes for linking the record into the search data structure, we get a total record length of 12 bytes, or 6 bytes per attached subnetwork. This results in an estimated router storage cost of $6 \times G_k$, that is 60–300 bytes, of record storage for each attached subnetwork, plus whatever overhead is required by the search data structure (i.e., hash table or search tree).

Finally, the cost to each subnetwork is the bandwidth consumed by the query and report packets. The number of packets per second is $(G_k + NJ_k + 1)/T_Q$, the same as the number of packet events that the routers must handle for that subnetwork. The queries and reports are small, fixed-length packets which, accounting for several layers of header, should not exceed 500 bits. Using our same estimates, that works out to approximately 125–500 bits per second of bandwidth consumed.

All of these costs are negligible for current-generation processors and subnetworks, and they remain insignificant even if the number of memberships per host or per subnetwork turns out to be an order of magnitude greater than our estimates, or if a smaller query interval $T_Q$ is chosen to speed up the detection of group disappearance by routers. The important scaling properties of the protocol are the following:

- Each host's costs are proportional only to the number of groups to which that host belongs, not to the numbers of members in those groups, nor to the total number of groups present on any subnetwork.

- Each router's costs are proportional only to the number of groups present on its attached subnetworks, not to the numbers of members in those groups.
- None of the protocol's costs are sensitive to the size of the internetwork.

## 5.3 Other Host Membership Protocol Issues

It would be possible to eliminate the query packets altogether—whenever a host sent or received a report packet for a particular group, it could simply reset its timer for that group to the interval $T_Q$, plus or minus a random amount of time between 0 and $T_D/2$. (There is no need for a query if the hosts can predict when the query would arrive!) However, the costs of the query packets are insignificant, and they have the advantage of leaving the choice of query interval up to the routers, thus avoiding the need to configure every host with an appropriate interval, and making it easier for network administrators to tune the interval in response to observed traffic loads.

Another advantage of using query packets is that, on an isolated subnetwork to which no routers (or no operational ones) are attached, there is no periodic reporting of membership—only the $N$ reports sent whenever a host joins a new group. This property might also be exploited by routers that are using the broadcast delivery approach (rather than truncated-broadcast), or on backbone subnetworks in some topologies—it enables the routers to "turn off" the periodic reports if they don't need them. The unsolicited reports sent when hosts join new groups could also be eliminated by having the hosts notice when they have not received any queries in a long time (e.g., $N$ times longer than the largest reasonable $T_Q$); this would only require hosts to maintain one more timer per attached subnetwork.

In the case where an interrogator is present on a subnetwork, a reduction in the number of unsolicited reports could be achieved by having each host listen to all reports, not just those for groups to which the host belongs. Then, when joining a group, a host would know if there are already members present and, if so, it could abstain from sending any unsolicited reports. This would require that reports be sent to the *all-hosts* group, rather than the reported group, and it would impose costs on the hosts similar to those imposed on the routers (that is, proportional to the

total number of groups present on the subnetwork, rather than just the number of groups to which the host belongs). However, as pointed out above, those costs are still modest, so this would be a worthwhile modification if the rate of joining groups becomes much higher than we anticipate.

In environments that require that hosts be billed for their memberships in host groups, the reports may be sent to an *all-routers* group, rather than the reported group or the *all-hosts* group. This prevents one host's membership report from being suppressed by another's, and enables the routers to keep track of the individual members of each group, at a corresponding increase in processing and storage cost to the routers and bandwidth cost to the subnetwork.

The Host Membership Protocol may operate at more than one layer in a protocol "stack". For example, a host may be attached to an extended LAN interconnected by IEEE 802 datalink-layer bridges, which is serving as one subnetwork in a DARPA IP-based internetwork. In that case, the host would send datalink-layer reports for its memberships in LAN multicast groups, as well as IP-layer reports for its memberships in IP host groups. The use of query packets ensures that the host will emit reports at all necessary layers (and that it doesn't continuously emit reports at unnecessary layers).

At layers where there is no hop-limit or time-to-live field in the packet header, as in IEEE 802 datalink-layer headers, the routers require some other way to prevent propagation of queries and reports beyond one hop. One other way is to simply recognize the query and report packet types as special cases that must never be forwarded. Existing LAN bridges that run the spanning-tree algorithm discussed in the next chapter already do this for their tree-building packets.

## 5.4 Chapter Summary

In this chapter, we have described a new protocol called the Host Membership Protocol, by which routers can learn what groups are present on their directly-attached subnetworks; routers need that information to know whether or not to forward any particular multicast packet onto those subnetworks. The protocol also serves the

important function of isolating hosts from any concern about the specific multicast routing algorithm that may be in use, and allowing that routing algorithm to be changed without any changes to the hosts. The overhead costs of the protocol—to hosts, routers, and the subnetwork itself—are negligible, and insensitive to the size of groups and the size of the internetwork.

It should also be noted that the design of the protocol respects the goal of keeping routers "stateless", which is one of the important characteristics of the datagram or connectionless approach to internetworking. In particular, the state that is required to ensure that a host stays in a group resides in the host itself, not in the routers; when a router fails and recovers, it learns anew of any group memberships, by asking the hosts. The state is lost only when the host fails, in which case the state is no longer needed anyway. (This property is what Clark calls "fate-sharing" [22]: the fate of the state information is tied to the fate of the entity that needs that state.) Also, since (as specified in the Host Group Model) the hosts are the "authorities" concerning their own memberships in groups, hosts can continue to join and leave groups, and to send and receive multicast on their own subnetworks, even when no routers are present or all attached routers have failed; this ensures that the multicast service works in the degenerate, but not uncommon, case of an internetwork containing only one subnetwork.

# Chapter 6

# Spanning-Tree Multicast Routing

The spanning-tree routing algorithm is used by many popular LAN bridges such as the DEC LANBridge 100 [35] and the Vitalink TransLAN [34], and it is being adopted as the standard routing protocol for interconnecting all types of IEEE 802 LANs at the datalink layer [39]. Unlike the routing algorithms we discuss in later chapters, the spanning-tree algorithm already supports multicasting across multiple subnetworks, using the broadcast delivery approach (that is, by delivering each multicast packet to every subnetwork). In this chapter, we describe extensions to the spanning-tree algorithm to reduce the cost and improve the scalability of the multicast service, by using either truncated-broadcast or multicast delivery.

## 6.1 Overview of Spanning-Tree (ST) Routing

The LAN bridge spanning-tree (ST) algorithm was first described by Perlman [55]; it has been further refined for use as an IEEE/ANSI Standard [4, 39]. Here, we describe those aspects of the ST algorithm necessary to understand our multicast extensions.

Under the ST algorithm, an arbitrary topology of routers and subnetworks[1] is

[1] We use the terms "router" and "subnetwork" rather than "bridge" and "LAN" for consistency with the abstract architectural model specified in Section 1.2, and to emphasize that the routing algorithm is largely independent of the layer(s) in which it is implemented in any particular internetwork. When this algorithm is used at the datalink layer, the set of interconnected LANs, sometimes called an "extended LAN", corresponds to what we call a single-level or "flat" internetwork.

converted into a loop-free topology by having some of the routers "ignore" their attachments to some subnetworks. The decision as to which subnetwork attachments to ignore is based on a distributed spanning-tree computation executed by all routers. By that computation, one router is elected *root* of the spanning tree. Then, for each subnetwork, one attached router is appointed *designated router*: for those subnetworks attached to the root, the root is appointed designated router; for all other subnetworks, the attached router that is the fewest hops from the root is appointed designated router (in the case of a tie, the one with the lowest address is appointed). Once these appointments have been made, each router is able to classify each of its attached subnetworks into one of the following three categories:

**Parent Subnetwork:** the attached subnetwork that is nearest the root (i.e., the fewest hops from the root). If there is more than one nearest subnetwork, one of them is arbitrarily chosen as parent. The root has no parent subnetwork; all other routers have exactly one parent subnetwork.

**Child Subnetwork:** any attached subnetwork for which this router is the designated router. Any router may have multiple child subnetworks.

**Ignored Subnetwork:** any other attached subnetwork. Any router may have multiple ignored subnetworks.

A router's attachments to its parent and child subnetworks constitute *branches* of the spanning tree, and all packet forwarding is restricted to those branches. The result is a loop-free topology, in which there is only one possible path between any pair of subnetworks.

The root periodically[2] transmits a special "tree maintenance" packet to all of its attached subnetworks, asserting its role as root. The packet is addressed to the *all-routers* multicast group, to which all of the routers belong; it is relayed to the other subnetworks via the designated routers. A failure of the root, a designated router, or a subnetwork is detected by the absence of that packet, which triggers a recomputation of (at least part of) the spanning tree. When a new router starts up,

[2]Typically, once a second.

it tries to claim the role of root, which also triggers a recomputation. Thus, topology changes may cause routers to reclassify their attached subnetworks as necessary to maintain a loop-free topology spanning all reachable subnetworks.

Routers listen to all packets transmitted on each of their attached subnetworks. Packets from ignored subnetworks are simply dropped. Packets from other subnetworks may be forwarded or dropped, depending on the contents of a local *routing cache*. The routing cache contains records of the form:

| host | branch | time-left |
|---|---|---|

where *host* is the address of a host, *branch* identifies which incident branch of the spanning tree leads to the host, and *time-left* is a timer used to detect stale cache records. When a router receives a packet from branch $k$ destined to host $h$, it looks up $h$ in its cache, and does the following:

- If there is a cache record for host $h$, and its associated branch is not the same as $k$, the router forwards the packet on that associated branch, thus sending it in the direction of the host.
- If there is a cache record for host $h$, and its associated branch is the same as $k$, the router drops the packet, because forwarding it on any other branch would be sending it in a direction known not to lead to the host.
- if there is no cache record for host $h$, the router does not know which direction leads to $h$; therefore, it forwards it on *all* branches except the arrival branch, $k$, to ensure that it reaches the host.

A router builds up its routing cache by observing the source addresses of incoming packets. When a packet from source $s$ arrives from branch $k$, the router looks up host $s$ in the routing cache. If there is no cache record for $s$, a record is created, with branch $k$ and a time-left value of $T_C$, which is the maximum lifetime of a cache entry in the absence of further traffic from the entry's host[3]. If a cache record already exists

[3] $T_C$ is typically on the order of a few minutes. It need only be short enough to ensure that a cache entry expires in less time than it normally takes to physically disconnect a host from one subnetwork and connect it to another.

for $s$, its branch is changed to $k$ if necessary, and its time-left value is reset to $T_C$. The expiration of a record's time-left causes that record to be removed from the cache. Whenever a topology change occurs, the time-left in each record is reduced to a very small value, to hasten its expiration if it has been invalidated by the change. (All routers are informed whenever the topology changes by the tree maintenance packets sent by the root.)

In most cases, any host that is being used as a destination of packets is also sending packets of its own (e.g., acknowledgements). Thus, the routers end up with cache entries for most hosts that are actively being used as destinations, so that they usually forward the packets only on those branches necessary to reach their destinations. Normally, only the first few packets sent to a previously inactive destination end up being forwarded on unnecessary branches (usually reaching all subnetworks).

Since multicast addresses are never used as source addresses, they are never added to the routing cache; a packet destined to a multicast address is always forwarded on all branches except its arrival branch[4]. This results in delivery of multicasts to all subnetworks, which is what we call the broadcast delivery approach.

The scale of an internetwork that uses ST routing is limited by several factors:

1. The storage required for each router's routing cache is proportional to the total number of active hosts in the internetwork. However, the amount of storage needed per cache record is so small (approximately 14 bytes, assuming 6-byte IEEE 802 addresses), that the other factors are much more likely to limit growth.

2. Since all packets are forwarded along the branches of a single tree, as the number of subnetworks and the traffic between them grows, the interior subnetworks of the spanning tree become bottlenecks; these bottlenecks cannot be relieved by providing alternative paths, only by increasing their capacity. (Alternative paths serve only as backup paths in case of failures in the spanning tree.)

[4]In some implementations, multicast address records may be manually configured into the router by system management, in order to prevent the propagation of particular multicast packets. We do not consider these records to be part of the routing *cache*, because they are permanent records and because they have different semantics than the unicast cache records—they simply cause packets destined to those addresses to be dropped in all cases, regardless of which branches they arrive on.

3. Since all multicast packets are forwarded to every subnetwork, the acceptable rate of multicasting by all hosts combined (or, alternatively, the acceptable number of hosts at a given per-host rate of multicasting) is limited by the capacity of the *lowest* capacity subnetwork in the internetwork, including leaf subnetworks. Furthermore, the unnecessary packet hops incurred by multicasts exacerbate the problem of the interior subnetworks becoming bottlenecks.

It is this third limit to scalability that is alleviated by our multicast routing extensions.

## 6.2 ST Truncated-Broadcast Routing

Truncated-broadcast delivery of a multicast packet means delivery of the packet along the branches of a broadcast tree rooted at the sender, but omitting the leaf subnetworks of the tree that have no members of the destination group. The ST routing algorithm already provides for delivery via a broadcast tree—that is how it currently handles multicast packets. All that is needed to truncate the tree is some way for the routers to identify leaf subnetworks, and to detect the presence or absence of destination group members on those subnetworks. The latter is accomplished by running the Host Membership Protocol described in Chapter 5. The former is not possible under the basic ST algorithm—although routers can identify their child subnetworks, they do not have enough information to know which of them are leaves of the spanning tree. However, this shortcoming is easily remedied, as described next.

### 6.2.1 Description of the ST Truncated-Broadcast Algorithm

In order for a router to know whether or not one of its child subnetworks is a leaf subnetwork, it needs to know if there is any other router that both (a) considers that subnetwork to be its parent and (b) itself has at least one child subnetwork. This can be seen from the example topology in Figure 6.1. In this example, router $r_1$ is the root of the spanning tree; each router is attached to its child subnetwork(s) below (i.e., downward in the figure) and to its parent subnetwork above. (Ignored attachments are not shown.) The three subnetworks below routers $r_2$, $r_3$, and $r_5$ are leaf subnetworks,

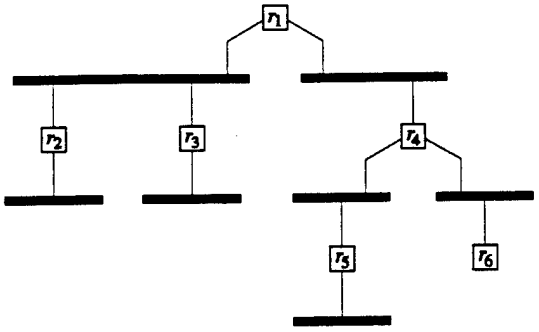



Figure 6.1: Example Topology for ST Routing

because none of them are the parent of any router. The subnetwork between $r_4$ and $r_6$ is also a leaf, even though it is the parent of $r_6$, because $r_6$ has no child subnetworks. (Router $r_6$ presumably has attachments to one or more other subnetworks, which are currently ignored in order to eliminate loops from the topology.)

If every router that had at least one child subnetwork were to report its presence on its parent subnetwork, the designated router for that parent subnetwork would be able to tell that it was not a leaf; leaf subnetworks could then be identified by the absence of any such reports. This reporting function is similar to that required to detect group presence, and can use exactly the same mechanism, as follows: if a router has any child subnetworks, it performs the *host* part of the Host Membership Protocol on its parent subnetwork, reporting its membership in the *all-routers* group on that subnetwork in response to the periodic membership queries from that subnetwork's interrogator.

The details can be seen in the pseudo-code of Figure 6.2. This algorithm operates on the same "group presence" and "attached subnetwork" objects used by the router part of the Host Membership Protocol (page 47), except that each attached subnetwork object $k$ has two additional attributes:

| | |
|---|---|
| $k$.status | current status of $k$: parent, child, or ignored |
| $k$.reporter | while true, report *all-routers* membership on $k$ |

Whenever the topology changes (event 1), the router reevaluates its responsibilities with respect to each of its attached subnetworks, based on their status as determined by the spanning-tree algorithm (the $k$.status attribute). For each child subnetwork,

```
1: on any topology change:
        for each attached subnetwork k
                k.interrogator ← (k.status = child)
                k.reporter ← (k.status = parent) and
                                (∃ subnetwork j such that j.status = child)

2: on receipt of multicast packet to nonlocal group g from subnetwork k:
        for each attached subnetwork j
                if j ≠ k and j.status ≠ ignored
                        if ∃ group presence p such that
                        (p.group = all-routers or p.group = g) and (p.subnet = j)
                                send copy of packet on subnetwork j
```

Figure 6.2: ST Truncated-Broadcast Algorithm

the router assumes the role of host membership interrogator. (That is, the router's election as a designated router under the ST algorithm serves also as an election for the role of interrogator, as required by the Host Membership Protocol.) Changing the state of $k$.interrogator invokes the procedures for being elected or resigning as interrogator on subnetwork $k$ (events 2 or 3 in Figure 5.3, page 48).

If the router has a parent subnetwork and at least one child subnetwork, it assumes the role of "reporter" on its parent subnetwork. Changing the state of the $k$.reporter attribute invokes the procedures for joining or leaving the *all-routers* group on $k$, in the host part of the Host Membership Protocol (events 1 or 2 in Figure 5.2, page 45).

Figure 6.3 shows the result of these decisions for the example topology. Each router's responsibilities with respect to each of its attached subnetworks is indicated by labels on the router's attachments: interrogator (I), reporter (R), or neither (no label). Also, every router performs the router part of the Host Membership Protocol on *all* of its attached subnetworks, including ignored subnetworks, so that if any attached subnetwork becomes reclassified as a child subnetwork as a result of a topology change, the router already knows what host groups are present on that subnetwork. (This means that a router may concurrently perform *both* the host part and the router part of the Host Membership Protocol on its parent subnetwork.)

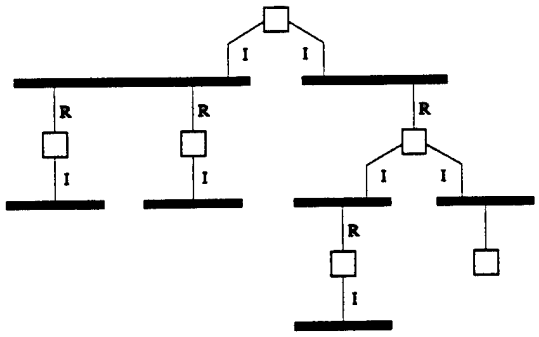



Figure 6.3: ST Router Responsibilities

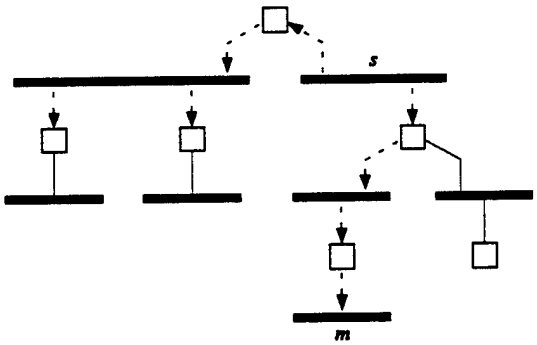



Figure 6.4: Example of ST Truncated-Broadcast Delivery

With the information provided by the Host Membership Protocol, a router performs truncated-broadcast delivery as shown by the pseudo-code for event 2 in Figure 6.2: each incoming multicast packet is forwarded to all subnetworks on which either a reporting router is present or the destination group is present, excluding the arrival subnetwork and any ignored subnetworks. Figure 6.4 shows the result of this forwarding strategy in the example topology, for a packet originating on the subnetwork labeled $s$, destined to a multicast group that has members only on the subnetwork labeled $m$. The packet traverses all interior subnetworks, but only the one leaf subnetwork on which destination members reside.

Event 2 does not include the reception of host membership queries, host membership reports, or spanning-tree maintenance packets; those packet types invoke separate procedures in the host membership or spanning-tree algorithms. It also does not apply to multicast packets whose hop limit has expired, in cases where the packets

have a hop-limit field (usually not true of datalink-layer packets); those are simply dropped.

### 6.2.2 Costs of the ST Truncated-Broadcast Algorithm

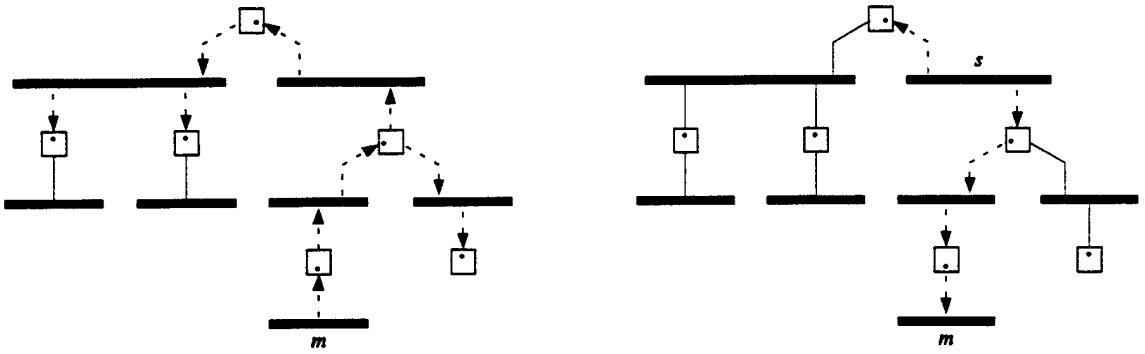
The costs of the truncated-broadcast extension to the spanning-tree algorithm are simply the costs of performing the Host Membership Protocol which are described in Chapter 5.

Most implementations of the ST algorithm employ highly-optimized software or special hardware to perform lookups in the routing cache; every unicast packet requires two lookups—one for the destination address and one for the source—and the speed of those operations determines the forwarding rate that can be sustained. The "group presence" records maintained by the Host Membership Protocol and referenced during multicast forwarding can and should use the same high-speed lookup mechanism—they are also indexed by address (multicast rather than unicast), and they are also critical to the speed of (multicast) forwarding. The group presence record for the *all-routers* group can be cached separately, to avoid having to look it up on every multicast reception.

The truncated-broadcast approach eliminates unnecessary packet hops onto the leaf subnetworks of the topology, thus freeing up bandwidth on those subnetworks and reducing the effect of leaf bandwidth as a constraint on the overall rate of multicasting. It still incurs excess packet hops on the interior subnetworks. The multicast delivery approach, described next, can be used to eliminate those excess interior hops, for topologies where the interior subnetworks have become bottlenecks.

## 6.3 ST Multicast Routing

In order to perform multicast delivery under the ST algorithm, each router needs to know which incident branches of the spanning tree lead to members of the packet's destination group. Routers are able to learn this type of information for *unicast* destinations by observing the source addresses of received packets—a host implicitly




(a) delivery of a membership report from *m* (b) delivery of a multicast packet from *s*

Figure 6.5: Example of ST Multicast Delivery with One Member Subnetwork

reports its relative location with every packet it sends. However, sending a packet conveys no information about what multicast groups the sender belongs to, so some other, explicit mechanism is required to report the relative locations of group members to the routers. Our multicast extension of ST routing builds upon the Host Membership Protocol, which reports group memberships to neighboring routers only, to provide internetwork-wide membership reporting.

### 6.3.1 Description of the ST Multicast Algorithm

Recall that, under the Host Membership Protocol, a host membership report packet is sent to the group that is being reported (in order to preempt reports from other members of the same group on the same subnetwork), and is hop-limited to one hop (to prevent propagation beyond the subnetwork.) Removing the hop limit would allow the report to propagate further, but would have the undesirable effect of preempting reports on other subnetworks. This effect can be avoided by having the routers attached to originating subnetwork (i.e., the neighboring routers of the reporting host) change the destination address of the report before forwarding it; if they change the destination to be the *all-routers* address, the packet can be delivered to all routers, without interrupting any hosts beyond those on the originating subnetwork.

Figure 6.5(a) illustrates the paths followed by such a report, using the same example topology as Figure 6.1. Assume there is a group $g$, which has one or more

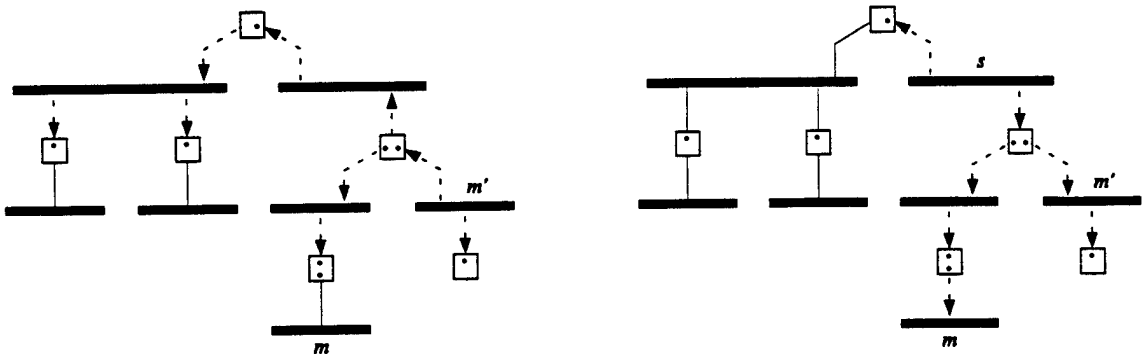
members on the subnetwork marked $m$. In response to each membership query from the router attached to $m$, one of the members transmits a host membership report with a destination address of $g$ and a hop-limit of one. When the attached router receives the report, it changes the destination address to *all-routers*, resets the hop limit to the maximum, and forwards a copy on every branch of the spanning tree, excluding the arrival branch and branches attached to leaf subnetworks[5]. As shown in the figure, this causes the report to be delivered to all routers, via all interior subnetworks.

When a router receives a forwarded report, besides forwarding it further, it treats it the same as a membership report received directly from a neighboring host. That is, it creates a "group presence" record indicating that the reporting group is present on the branch from which the report was received; if a record already exists, its "time-left" timer is refreshed. (Instead of a group presence record indicating presence only on an attached subnetwork, it now indicates presence anywhere on that subnetwork or *beyond*, the same as routing cache entries do for individual hosts.) In the example of Figure 6.5(a), the branch that each router associates with the group $g$, as a result of the report from $m$, is indicated by a dot inside the router.

With this information in the routers, multicast delivery is accomplished by having each router forward a received multicast packet only on those branches that lead to the destination group, as indicated by the router's group presence records. This is illustrated in Figure 6.5(b): a multicast packet originating on the subnetwork marked $s$, destined to the group whose only members are on subnetwork $m$, is forwarded by the routers "in the direction of the dots," thus traversing only those subnetworks required to reach the group members.

Figure 6.6(a) shows what happens when members of the group appear on a second subnetwork, $m'$. The membership report from $m'$ is delivered to all routers, causing them either to add a new branch for the group (additional dots in the figure), or to refresh their previous knowledge of the group's presence on a branch. Figure 6.6(b)

[5]The routers use the same leaf-detection scheme as described for truncated-broadcast delivery, except that every router reports its membership in the *all-routers* group on its parent subnetwork, whether or not it has any child subnetworks. This ensures that all routers receive copies of all reports.




(a) delivery of a membership report from $m'$ (b) delivery of a multicast packet from $s$

Figure 6.6: Example of ST Multicast Delivery with Two Member Subnetworks

shows how a multicast packet from $s$ would be delivered, when there are members on both $m$ and $m'$.

The algorithm as described so far does accomplish delivery via a multicast tree. However, as we pointed out in an earlier paper [26], it requires every interior subnetwork to carry the total membership reporting traffic from all subnetworks. Although the membership traffic from any one subnetwork is expected to be very low (see Chapter 5), the summation of that traffic presents a potential scaling problem, especially given the tendency of the interior subnetworks to become bottlenecks under the ST routing scheme. Fortunately, this problem is easily alleviated: instead of a router immediately forwarding received membership reports, it can delay for a while, accumulating a number of reports, and then forward them all in a single packet containing a list of group addresses (and with a hop-limit of one). A single delayed report can take the place of individual reports from different groups, as well as reports from the same group on different subnetworks. (An example of the latter can be seen by comparing Figures 6.5(a) and 6.6(a): the router above $m'$ can send a single report for group $g$ in the direction towards the root, in place of the two reports from $m$ and $m'$.)

A router need not explicitly store reports while waiting to forward them: at regular reporting intervals $T_R$, it can simply examine its set of group presence records to see which ones have been refreshed within the last $T_R$ interval, and send the addresses of those groups in a new report. An appropriate value for the reporting interval $T_R$

is $T_Q + T_D$, which is the host membership query interval plus the maximum random delay of a host membership report—within that interval, all existing groups should be heard from, at least once (in the absence of delivery errors).

The final algorithm is shown in Figure 6.7; the statements enclosed in <> brackets are invocations of the procedures listed in Figure 6.8. This algorithm operates on the same "group presence" and "attached subnetwork" objects as the truncated-broadcast algorithm, except that each attached subnetwork object $k$ has two additional attributes:

| | |
|---|---|
| $k$.rtimer | time till next report is sent on $k$ |
| $k$.rcount | retransmission counter for reports on $k$ |

On a topology change (event 1), the router reevaluates its roles with respect to each attached subnetwork, as explained for truncated-broadcast routing[6], and it initializes its report timer for each non-ignored subnetwork. Also, when the topology changes, the router's previous knowledge of group presence may become invalid: group members that were previously reachable in one direction may now be reachable in a different direction, relative to this router. This is handled by assuming that all known groups have become reachable in *all* directions, and updating their group presence records accordingly. This has the effect of reverting to broadcast delivery for an interval of $T_P$ after any topology change.

The pseudo-code for event 2 replaces the corresponding code in the router part of the Host Membership Protocol (event 5 in Figure 5.3, page 48). Reports may be received either from neighboring hosts or from neighboring routers; reports from hosts contain only a single group address, whereas reports from routers may contain multiple addresses.[7] If the incoming report lists any groups that were not previously known to be present on the arrival branch (indicated by the global variable "newgroup" in the pseudo-code), the router immediately generates a report on all other branches, and sets the timer and counter for those branches to trigger $N - 1$ more

[6]Note that, unlike the truncated-broadcast case, the router does not need to have a child subnetwork before assuming the role of reporter on its parent subnetwork.

[7]If a router has more group addresses to report than fit in a single packet, it sends as many packets as necessary. The handling of that case is not included in the pseudo-code.

```
1: on any topology change:
        for each attached subnetwork k
                k.interrogator ← (k.status = child)
                k.reporter ← (k.status = parent)
                k.rtimer ← (k.status = ignored) ? 0 : T_R
                k.rcount ← N
                for each group presence p
                        <record p.group presence on subnetwork k>

2: on receipt of report packet from subnetwork k:
        new-group ← false
        for each group address g in the report packet
                <record group g presence on subnetwork k>
        if new-group
                for each attached subnetwork j
                        if j ≠ k and j.status ≠ ignored
                                <send report to subnetwork j>
                                j.rtimer ← T_N; j.rcount ← 2

3: on expiration of k.rtimer for subnetwork k:
        <send report to subnetwork k>
        if k.rcount < N
                k.rtimer ← T_N; k.rcount ← k.rcount + 1
        else
                k.rtimer ← T_R

4: on receipt of multicast packet to nonlocal group g from subnetwork k:
        for each attached subnetwork j
                if j ≠ k and j.status ≠ ignored
                        if ∃ group presence p such that
                        p.group = g and p.subnet = j
                                send copy of packet on subnetwork j
```

Figure 6.7: ST Multicast Algorithm

```
<record group g presence on subnetwork k>:
        if ∃ group presence p such that p.group = g and p.subnet = k
                if p.time-left ≤ T_P − T_R
                        new-group ← true
                p.time-left ← T_P
        else
                create new group presence p; new-group ← true
                p.group ← g; p.subnet ← k; p.time-left ← T_P

<send report to subnetwork k>:
        if ∃ group presence p such that p.group = all-routers and p.subnet = k
                create empty report packet
                for each group presence p such that
                        p.subnet ≠ k and p.subnet.status ≠ ignored and
                        p.time-left > T_P − T_R
                                add p.group to report packet
                if report packet not empty
                        send report packet to all-routers on subnetwork k
```

Figure 6.8: ST Multicast Algorithm—Supplemental Procedures

reports at short intervals $T_N$.[8] This ensures that the new branch is quickly and reliably added to the group's multicast tree, to minimize join latency. Also, if a router receives a report for a group whose time-left is not within $T_R$ of $T_P$, the presence timeout, it means that no previous report was received for that group in the preceding $T_R$ interval; it indicates either that a report was lost or that a host has just joined the group shortly after all previous members on the same branch have left the group. This situation is treated the same as a new group appearance.

When a subnetwork's report timer expires (event 3), a report packet is constructed for that subnetwork, listing all groups that have been heard from on any *other* subnetwork, in the last $T_R$ interval. After sending the report, the router resets the timer to either the short interval $T_N$, as discussed in the previous paragraph, or to the regular reporting interval $T_R$.

Finally, incoming multicast packets (event 4) are forwarded to all subnetworks over which the destination group is reachable, excluding the arrival subnetwork and any ignored subnetworks. (This event does not include the reception of host membership queries, host membership reports, spanning-tree maintenance packets, or any multicast packets whose hop-limit has expired.)

### 6.3.2 Costs of the ST Multicast Algorithm

The multicast extension to the ST algorithm incurs the following costs, beyond those of the Host Membership Protocol described in Chapter 5:

- The storage cost of keeping group presence records for all existing groups in all routers.
- The bandwidth cost of conveying router-to-router report packets across each interior subnetwork.
- The computational cost to the routers of generating and processing the router-to-router report packets.

[8] $N$ and $T_N$ are defined in Chapter 5.

To estimate the storage cost, we assume that routers maintain group presence records of the form suggested in Chapter 5, that is:

| group | time-left$_1$ | ... | time-left$_n$ | link |
|---|---|---|---|---|

where $n$ is the number of attached subnetworks. For a router with two subnetwork attachments[9], operating at the datalink layer with 6-byte IEEE 802 addresses, such records would be 12 bytes long (assuming 2-byte timers and a 4-byte pointer to link the record into the search data structure). This means that four kilobytes of storage would be sufficient to hold more than 300 group presence records, which is more than the number of groups we would expect in any internetwork based on ST routing.

(Another way to look at the storage requirement is in comparison with that required to store unicast routing cache records. One popular LAN bridge [35] provides space to hold cache records for up to 8000 active hosts, each of which requires about 14 bytes. Space for a few hundred group presence records would not be a significant additional cost.)

Each interior subnetwork must carry the router-to-router report packets generated by its attached routers. These may be divided into two types: periodic reports sent by every attached router at intervals of $T_R$, and sporadic reports caused by new groups appearing or existing groups gaining members on new branches.

The periodic reports are expected to consume very little bandwidth on a subnetwork, since the reporting interval should be relatively long. (The assumed values for $T_Q$ and $T_D$ from Table 5.1, page 51, yield a $T_R$ of $50 + 10 = 60$ seconds.) Each report sent by an attached router lists the addresses of all groups present in the direction(s) away from the subnetwork. (This can be visualized as illustrated in Figure 6.9. An interior subnetwork has a set of attached routers, each of which leads to a separate "cloud" of subnetworks, that is, a subtree of the global spanning tree. The report packets sent by each router on the interior subnetwork contain the addresses of all groups present in that router's cloud.) We would expect the list of addresses to fit in a single packet—an Ethernet packet, which has the smallest maximum packet size of any IEEE 802 LAN, can hold 250 6-byte addresses—so the periodic report cost

[9]Most datalink-layer bridges have only two LAN attachments.

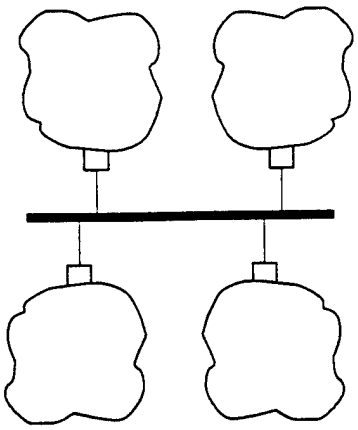


Figure 6.9: An Interior Subnetwork in an ST Routing Environment

is simply $R_k/T_R$ packets per second, where $R_k$ is the number of routers attached to subnetwork $k$. Ten attached routers, reporting 250 addresses each at intervals of 60 seconds would generate only 0.15 packets per second, consuming 2000 bps of bandwidth (.02% of the bandwidth of an Ethernet).

The bandwidth cost of sporadic reports is harder to anticipate. Referring again to Figure 6.9, whenever a new group appears in one of the clouds, it causes $N$ reports to be transmitted on the interior subnetwork. The frequency of such reports depends not only on how often new groups appear in the internetwork, but also on the distribution of members of those groups, relative to any interior subnetwork: a group with a single member in every cloud will incur $R_k \times N$ reports, whereas a group with any number of members (on any number of subnetworks) within a single cloud will generate only $N$ reports. In any case, we do not expect this reporting load to be significant, based on the behavior of existing and anticipated multicast applications—the average rate of appearance of new groups should be very low, relative to the capacity of the subnetworks to carry their reports. (If that should prove not to be the case in a particular internetwork, it would be straightforward to implement a limit, adjustable by network management, on the number of new group reports that may be sent by any router within a given time interval. That would allow the report traffic bandwidth to be bounded, at the cost of increased join latencies during episodes of rapid new group appearance.)

Since the frequency of router-to-router reports is expected to be very low, the computational cost to the routers of generating and processing them is expected to be correspondingly low. To generate a report, a router must scan through all of its group presence records, but that requires only a few instructions per record, and there are not likely to be more than a few hundred records. Processing an incoming report involves updating all records corresponding to the groups listed in the report—the same high-speed lookup mechanism used for routing unicast and multicast packets can be used for quickly locating the records to be updated.

Finally, we have discussed the group presence records as if they were separate from the unicast routing records. However, the two types of records essentially serve the same purpose: they tell the router which subnetwork(s) to use to forward a packet towards its destination(s). Both types need to be quickly accessible by address, and both types contain timers for detecting stale information. They should be able to share the same procedures or mechanisms for allocation, access, and reclamation. The only significant differences are:

- group presence records are updated by explicit report packets, whereas unicast routing records are updated by recording the source addresses of all incoming packets[10],

- group presence records can point to more than one attached subnetwork, with a separate timer for each, whereas unicast routing records point to only one subnetwork, with a single timer, and

- the router behavior when there is no record for the destination of an incoming packet is different for multicast and unicast destinations: a multicast packet is dropped, while a unicast packet is forwarded to all attached subnetworks.

[10]In an earlier description of the ST multicast algorithm [26], we suggested that a membership report should carry the address of the reporting group in its source address field, so as to make the processing of group address updates as similar as possible to the processing of unicast address updates. Unfortunately, in the IEEE 802 bridge environment for which this algorithm is mainly intended, the bit that indicates that an address is a group address has been usurped for a different, incompatible purpose, when used in the source address field of a packet [33].

## 6.4 Chapter Summary

In this chapter, we have described extensions to the spanning-tree routing algorithm to support truncated-broadcast or multicast delivery. The algorithms satisfy the reliability and performance requirements of the Host Group Model, by ensuring that multicast packets follow the same paths as unicast packets, on their way to each destination group member. The overhead costs of the algorithms are very small, relative to the resources available and the overhead of unicast routing, and those costs are proportional only to the number of groups present on a subnetwork (in the case of truncated-broadcast routing) or present in the internetwork (in the case of multicast routing). (For comparison, the overhead of unicast routing is proportional to the number of active hosts in the internetwork, which would normally be greater than the number of groups; hence, the multicast routing algorithm scales at least as well as the unicast routing.)

The use of these algorithms can eliminate unnecessary packet hops incurred by the broadcast strategy by which multicast packets are delivered in existing spanning-tree internetworks. The bandwidth thus freed up becomes available for productive traffic, allowing better service for existing hosts or more capacity for growth.

# Chapter 7

# Distance-Vector Multicast Routing

The distance-vector routing algorithm, also known as the Ford-Fulkerson [31] or Bellman-Ford [7] algorithm, has been used for many years in many networks and internetworks. For example, the original Arpanet routing protocol [51] was based on distance-vector routing, as was the Xerox PUP Internet [12] routing protocol. It is currently in use in many parts of the DARPA Internet and in private internetworks based on the DARPA protocols or on other, proprietary protocols such as AppleTalk [61] or XNS [67]. One well-known and widely-used implementation of distance-vector routing is the *routed* program available on all BSD (and related) UNIX systems [36].

## 7.1 Overview of Distance-Vector (DV) Routing

A router that uses the distance-vector (DV) algorithm maintains a routing table that contains an entry for every reachable destination in the internetwork, where a "destination" may be a single host, a single subnetwork, or a cluster of subnetworks. The routing table entry for each destination contains the following fields:

| destination | distance | next-hop-address | next-hop-subnet | time-left |
|---|---|---|---|---|

*Distance* is the shortest-path distance to the destination, typically measured in hops or some other unit of delay. *Next-hop-address* is the address of the next router on

the path towards the destination, or the address of the destination itself if it shares a subnetwork with this router. *Next-hop-subnet* is a local identifier of the subnetwork used to reach *next-hop-address*. *Time-left* is a timer used to detect stale route entries, that is, entries for destinations that have become unreachable.

The router initializes its routing table with entries for those destinations that are directly reachable on its attached subnetworks, and augments it with information learned from other routers attached to those subnetworks. In particular, the router periodically[1] receives a packet of the following form from each of its neighboring routers:

| $\text{destination}_1$ | $\text{distance}_1$ | $\text{destination}_2$ | $\text{distance}_2$ | ... |
|---|---|---|---|---|

listing all destinations known to the sending router, along with their distances from that router. (This is the "distance vector" from which the algorithm gets its name.) By comparing the contents of these packets with its own table, the receiving router can learn of other destinations and of the best next-hop routers to reach those destinations. This information is used to update the receiving router's table and is, in turn, reported in this router's periodic routing packets; the packets are sent as a single-hop multicast or broadcast on each of the router's attached subnetworks so as to reach all of its neighbors. By this neighbor-to-neighbor exchange, all routers eventually end up with routing tables identifying all reachable destinations in the internetwork, along with a *next-hop-address* for the shortest path to each of those destinations.

Most of the details of how the routing table is updated, how the timers are handled, and how the periodic reports are generated are not relevant to the multicast extensions we describe below; those details can be found in the references cited above. However, one feature that is relevant to the following discussion is the so-called "split horizon with poisoned reverse" optimization [36] implemented in some versions of the DV algorithm to speed up routing table convergence after a topology change. This optimization works as follows: When a router generates a routing packet to be sent on attached subnetwork $k$, for all those destinations whose *next-hop-subnet* equals $k$, the router reports a distance of "infinity" rather than the distance recorded in the

[1] Typically, once every 10–60 seconds.

```
on receipt of broadcast packet from source s via link k:
    if ∃ route r such that r.destination = s and r.next-hop-link = k
        for each incident link j
            if j ≠ k
                send copy of packet on link j
```

Figure 7.1: The Reverse-Path Forwarding Algorithm of Dalal and Metcalfe

router's table. This ensures that no other routers on subnetwork $k$ will choose this router as their first hop to reach those particular destinations; such a choice would clearly be incorrect, since this router would simply forward any packets for those destinations back onto subnetwork $k$.[2] In the following discussion of multicast routing, we assume that this optimization is implemented; it is trivial to add to any version DV routing that does not already support it.

## 7.2 DV Truncated-Broadcast Routing

Our algorithm for DV truncated-broadcast routing is a refinement of Dalal and Metcalfe's reverse-path forwarding (RPF) algorithm [24]. The RPF algorithm performs *broadcast* delivery in a store-and-forward network of point-to-point links, using only the information present in typical routing tables such as those maintained by DV routing.

A pseudo-code representation of the basic RPF algorithm is shown in Figure 7.1. It operates on "route" objects, which are the entries in a routing table maintained by any shortest-path unicast routing algorithm. Each route $r$ has two attributes of relevance to the RPF algorithm:

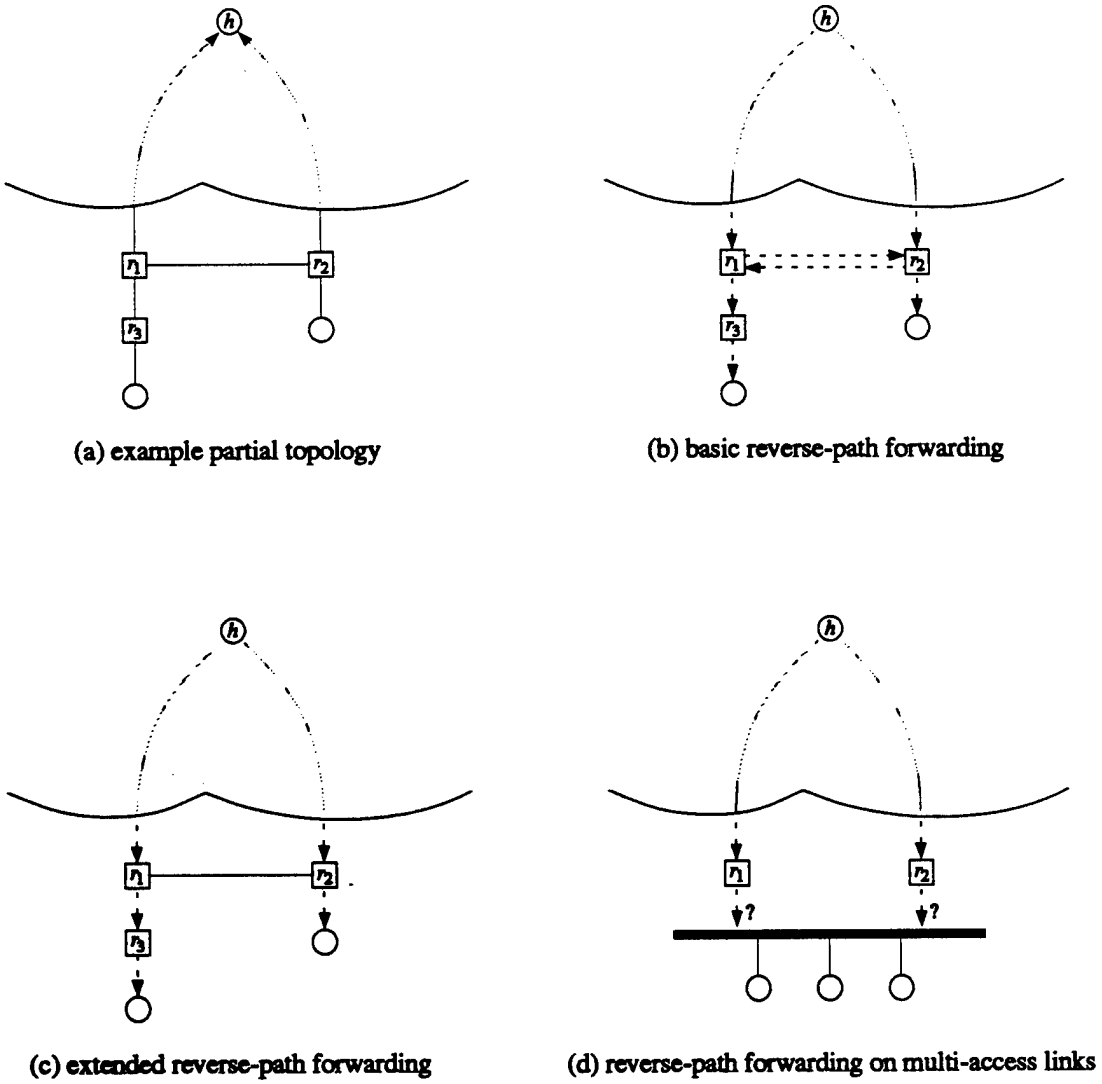
[2] Without this optimization, such erroneous choices—which can occur only as a result of a topology change—are eventually corrected by the normal operation of the algorithm. However, the routers may take considerably longer to reach the right choices, during which time some packets may be caught in a forwarding loop, unable to reach their destinations. Even with the optimization, there are still some (less likely) circumstances under which temporary forwarding loops may be formed, involving routers on different subnetworks. This is a well-known and widely-studied problem with DV routing; a good explanation is provided by Hedrick [36].

| | |
|---|---|
| $r$.destination | a destination host address |
| $r$.next-hop-link | identifier of next link on shortest path to $r$.destination |

As can be seen from the pseudo-code, the basic RPF algorithm is very simple: if a broadcast packet arrives via the link that would be used to reach the source of the packet, a copy of the packet is forwarded on all other incident links; if the packet arrives via any other link, it is ignored. This accomplishes delivery of the packet to every node (i.e., host or router) in the network. Each node receives a copy of the broadcast packet over the reverse of the path used to send unicasts to the source of the broadcast, which is why it is called reverse-path forwarding. In the normal case where the length of each path is the same in both directions (such as when using hop count as the distance measure) RPF accomplishes shortest-path delivery, which satisfies the service requirements of our Host Group Model.

One drawback of the basic RPF algorithm is that it generates more packet copies than necessary. Consider the partial topology illustrated in Figure 7.2(a). In this example, three routers and two hosts are interconnected by point-to-point links; they are also part of a larger network (whose details are obscured behind a cloud). The two dotted lines represent the shortest paths to a particular host $h$ somewhere in the network, from routers $r_1$ and $r_2$ respectively. Figure 7.2(b) shows how the routers forward a broadcast packet originating at $h$, under the basic RPF algorithm. Each router receives a copy of the packet via the link it uses to reach $h$, and forwards a copy to every other incident link. The two packet copies sent on the link between $r_1$ and $r_2$ are superfluous—that link does not belong to the shortest reverse-path tree rooted at $h$. Those excess packet copies are simply ignored by the receiving routers, since they do not arrive on the links used to reach $h$.

To eliminate the excess packet copies, Dalal and Metcalfe suggest having each router periodically send a message to each of its neighboring routers, saying "I use this link to reach these destinations". For example, in our topology, $r_3$ periodically sends a message on its link to $r_1$, saying that it uses that link to reach host $h$ (and all other hosts except the one directly connected to $r_3$). Then, when a router forwards a broadcast packet, rather than sending copies on all links except the incoming link, it sends copies on only those links that either lead to a host or lead to a router that




(a) example partial topology

(b) basic reverse-path forwarding

(c) extended reverse-path forwarding

(d) reverse-path forwarding on multi-access links

Figure 7.2: Reverse-Path Forwarding Example

claims to use the link to reach the source of the broadcast. As seen in Figure 7.2(c), this extended RPF algorithm eliminates the superfluous packet copies. Unfortunately, this extension does not work in an internetwork in which the routers may be joined by multi-access links (subnetworks), as well as point-to-point links. This can be seen in Figure 7.2(d): if routers $r_1$ and $r_2$ treat the multi-access link as a host link, they both forward a copy of a broadcast packet from $h$ onto that link, resulting in one excess copy; if they treat it as a router link, neither sends a copy (since no router uses the link to reach $h$), and the broadcast fails to reach all hosts. Our truncated-broadcast algorithm provides a new solution to the RPF excess-packet problem, suitable for a distance-vector-based internetwork that contains multi-access links. It also handles the more general problem of delivering multicast rather than broadcast packets.

### 7.2.1 Description of the DV Truncated-Broadcast Algorithm

Our DV truncated-broadcast algorithm is shown in Figure 7.3. It is used in conjunction with a normal DV routing protocol and with the router part of the Host Membership Protocol (see Chapter 5). The algorithm operates on four types of object: "route", "neighbor", "group presence", and "attached subnetwork". Route objects are the entries in the DV routing table; a route object $r$ has the following attributes:

| | |
|---|---|
| $r$.destination | internetwork destination address |
| $r$.distance | shortest-path distance to destination |
| $r$.next-hop-subnet | next-hop subnetwork towards destination |
| $r$.child[$k$] | child flags, per attached subnetwork $k$ |
| $r$.leaf[$k$] | leaf flags, per attached subnetwork $k$ |

The first three attributes are a subset of the fields normally present in a DV routing table entry. The $r$.child and $r$.leaf attributes are additional fields required by our algorithm; their purpose and use is described below. Neighbor objects represent the information received in the most recent DV routing packet from each neighboring router; a neighbor object $n$ has the following attributes:

```
1: on change of router population on subnetwork k:
        k.interrogator ← (∃ no neighbor n such that
                          n.subnet = k and n.address < k.my-address)

2: on change of information about destination d from subnetwork k:
        if ∃ route r such that r.destination = d
            r.child[k] ← (r.next-hop-subnet ≠ k) and
                         (∃ no neighbor n such that n.subnet = k and
                              (n.distance[d] < r.distance or
                              (n.distance[d] = r.distance and
                              n.address < k.my-address)))
            r.leaf[k] ← (∃ no neighbor n such that
                         n.subnet = k and n.distance[d] = infinity)

3: on receipt of multicast packet from source s to group g via subnetwork k:
        if ∃ route r such that r.destination = s and r.next-hop-subnet = k
            for each subnetwork j
                if r.child[j] and
                ((not r.leaf[j]) or (∃ group presence p such that
                    p.group = g and p.subnet = j))
                        send copy of packet on subnetwork j
```

Figure 7.3: DV Truncated-Broadcast Algorithm

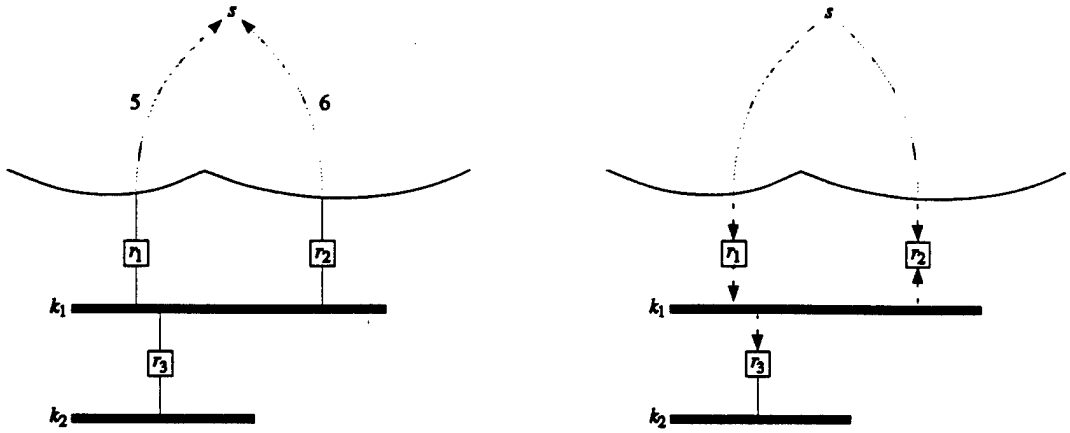
| | |
|---|---|
| $n$.subnet | subnetwork to which neighbor is attached |
| $n$.address | neighbor's address on subnetwork $n$.subnet |
| $n$.distance[$d$] | distances from neighbor to each destination $d$ |

Group presence objects and attached subnetwork objects are as defined for the router part of the Host Membership Protocol (page 47), except that each attached subnetwork object $k$ has one additional attribute:

| | |
|---|---|
| $k$.my-address | this router's address on subnetwork $k$ |

The Host Membership Protocol requires that one router on each subnetwork be elected as the membership interrogator. Under our DV algorithm, the router with the lowest address on the subnetwork arbitrarily assumes that role. This is handled as shown in the pseudo-code for event 1, which executes whenever a new router appears on the subnetwork (i.e., either this router starts up, or a routing packet is received from a new neighboring router), or a router disappears from the subnetwork (i.e., the time since receiving the last routing packet from a neighboring router exceeds some limit). Changing the state of the $k$.interrogator attribute invokes event 2 or 3 in Figure 5.3 (page 48).

The problem identified above, of excess packets when using RPF over multi-access links, is solved as follows: on each subnetwork $k$, the attached router that has the shortest distance back to a particular multicast source $s$ assumes sole responsibility for forwarding multicast packets from $s$ onto subnetwork $k$; in the case of a tie, the router with the lowest address (arbitrarily) wins. For example, consider the topology illustrated in Figure 7.4(a). Assume that hop count is used as the measure of path length, and that the shortest-path distances from routers $r_1$ and $r_2$ to a particular multicast source $s$ are 5 and 6 hops, respectively; $r_3$'s distance to $s$ is 6 hops, via $r_1$. Of the three routers attached to $k_1$, $r_1$ has the shortest distance to $s$; therefore, $r_1$ assumes responsibility for forwarding multicasts from $s$ onto $k_1$. On $k_2$, that responsibility is assumed by the one attached router, $r_3$. We say that $k_1$ is the *child* of $r_1$, and that $k_2$ is the child of $r_3$, in the shortest-reverse-path tree rooted at $s$. (This is the same method as that used by the ST routing algorithm described in Section 6.1 to organize




(a) example partial topology (b) truncated-broadcast delivery

Figure 7.4: Example of DV Truncated-Broadcast Delivery

the routers into a single, loop-free broadcast tree, except that in this case the routers form multiple trees, one for each potential multicast source.)

Each router is able to determine which of its attached subnetworks are its child subnetworks for each possible source, by comparing its own distance to the source with the distances periodically reported by all of its neighbors. The first assignment statement in the pseudo-code for event 2 shows how this determination is made: a subnetwork is a child for $s$ if it is not the next-hop subnetwork towards $s$ and if there is no other router on the subnetwork with a shorter distance to $s$ (or with the same distance and a lower address). For each entry in the routing table, i.e., each possible source of multicasts, the set of child subnetworks is recorded in the $r$.child attribute.

The router is also able to tell from the periodic routing packets which of its attached subnetworks are *leaf* subnetworks of the reverse-path trees for each source. A leaf subnetwork is one that no router uses to reach the source. (For example, in Figure 7.4(a), subnetwork $k_2$ is a leaf of $s$'s reverse-path tree.) Assuming the routers use the "split horizon with poisoned reverse" optimization, described previously, any router that uses a subnetwork $k$ to reach a source $s$ reports a distance of infinity to $s$, in its routing packets sent on $k$. The absence of any such reports indicates that a subnetwork is a leaf for $s$; this is recorded by the router in the $r$.leaf attribute of $s$'s

route entry, as shown in the second assignment statement under event 2.

The determination of the child and leaf status of a subnetwork $k$, relative to a source $s$, is performed whenever the router's distance or next-hop-subnetwork for $s$ changes, or whenever the distance to $s$ reported by any other router on subnetwork $k$ changes. Event 2 in the pseudo-code refers to any of these occurrences.

Finally, truncated-broadcast delivery of multicast packets is accomplished as shown by the pseudo-code for event 3. In order to be forwarded, a multicast packet must arrive via the next-hop subnetwork used to reach the source of the packet, which is the basis of reverse-path forwarding. If that requirement is satisfied, a copy is forwarded to all child subnetworks for that source that are not leaf subnetworks or that have members of the destination multicast group.

Figure 7.4(b) shows the result of this forwarding algorithm in our example partial topology, assuming there are no members of the destination group on $k_2$. (The presence or absence of members on $k_1$ does not affect the outcome, since $k_1$ is not a leaf of $s$'s reverse-path tree.) Note that router $r_2$ receives two copies of the packet. The copy from $k_1$ is discarded because it arrives on the "wrong" subnetwork for multicasts from $s$. The copy that arrives on the "right" subnetwork is also discarded, because $r_2$ has no child subnetworks for multicasts from $s$. Router $r_3$ discards its copy because its one child subnetwork is a leaf for multicasts from $s$, on which no destination group members are present.

### 7.2.2 Costs of the DV Truncated-Broadcast Algorithm

In addition to the negligible costs of performing the Host Membership Protocol, shared by all of our algorithms, the DV truncated-broadcast algorithm incurs the storage cost of the child and leaf information added to each entry in the routing table, and the computational costs of maintaining that information. The child and leaf fields are simply two bit maps, with a bit for each attached subnetwork; two additional bytes per route entry would be sufficient to support up to 8 attached subnetworks, and would increase the total size of the routing table by less than 15% (assuming a normal routing entry is at least 16 bytes long). The child and leaf fields must be updated only when the distances to particular destinations change due to the

failure or recovery of a router or a subnetwork, which are infrequent occurrences in most internetworks. In any case, the cost to update the child and leaf information for each route entry is small and proportional only to the number of neighboring routers on a single subnetwork; in most implementations, the overhead of simply receiving and decoding the periodic routing packets from those neighbors far outweighs the additional cost of maintaining the child and leaf information.

We have assumed that the router keeps a copy of the most recent routing packet received from each neighbor. However, some older implementations of the distance-vector routing algorithm do not do so—each packet is used to to update the routing table and then immediately discarded. Our algorithm can be modified to operate in a similar "incremental" manner, at the cost of possible duplication of multicast packets for a short time after any topology change, and some additional space in each route entry.[3] However, the memory savings achieved by not saving the most recent routing packets do not seem to warrant the extra complexity of the incremental approach and the cost of the duplicate multicast packets, given today's low cost of memory. Furthermore, modern implementations of DV routing do store the most recent information from each neighboring router, to support other features[4].

[3]The procedure is to have each router initially assume that, for each source $s$, all attached subnetworks except the one used to reach $s$ are child subnetworks. If a routing packet arrives from a neighbor on one of those subnetworks $k$, reporting a shorter distance to $s$, this router relinquishes its claim of $k$ as a child, but remembers the address of the router that has reported the shorter distance; that router is considered a *dominant* router on subnetwork $k$ for source $s$. If that dominant router subsequently fails, or reports a distance to $s$ larger than this router's distance to $s$, this router (and, perhaps, other routers attached to the same subnetwork) again adopts the subnetwork as a child. During the interval when more than one router considers the subnetwork as a child, excess copies of multicast packets may be forwarded onto the subnetwork. A similar approach is taken to handling the leaf information: a router considers a subnetwork to be a leaf until some other router on that subnetwork reports a distance of infinity for $s$; such a router is considered a *subordinate* router on subnetwork $k$ for source $s$. When that subordinate router fails, or reports a non-infinite distance, the first router waits a short period of time to learn if there are any other routers reporting an infinite distance and, if not, again assumes that the subnetwork is a leaf. To support this version of the algorithm, it is necessary to store, in each routing entry, the address of a dominant router for each subnetwork (if other than this router) and the address of a subordinate router on each subnetwork (if there is one). These addresses replace the child and leaf bit maps in the routing entries.

[4]Examples include multipath routing (that is, routing to a given destination via more than one path, to improve throughput or link utilization) or fallback routing (that is, pre-computing fallback routes to be used when current routes fail, for faster recovery and avoidance of routing loops when the topology changes).

Our DV truncated-broadcast algorithm has no bandwidth overhead—the normal DV routing packets convey all the information needed by the algorithm. Of course, the truncated-broadcast delivery strategy generally consumes more bandwidth when delivering a multicast packet than does the multicast delivery strategy. A refinement of our truncated-broadcast algorithm to provide true multicast delivery is described next.

## 7.3 DV Multicasting Routing

A multicast delivery tree may be thought of as a broadcast delivery tree that has been pruned back so that it does not extend beyond those subnetworks that have members of the destination group. In an internetwork using shortest-path routing, there is potentially a different multicast tree for every combination of source and group in the internetwork, so the cost of pre-computing all possible multicast trees scales poorly as the size of the internetwork (in terms of number of possible multicast sources) and the number of host groups increases. However, we would not expect every source to be sending to every group at the same time—the set of *active* multicast trees should be much smaller than the set of potential trees, over some interval. In this section, we describe an algorithm that exploits that characteristic, to provide efficient multicast delivery in a distance-vector routing environment. The algorithm is based on the idea of *on-demand pruning* of multicast trees, in which the routing costs associated with a particular multicast tree are incurred only while that tree is active, that is, when there is multicast traffic being delivered along the tree.

### 7.3.1 Description of the DV Multicast Algorithm

When a multicast packet is sent from a particular source to a particular group, and the routers have no previous information about that (source, group) pair, the packet is delivered according to the truncated-broadcast algorithm described in the previous section. When the packet reaches a router for whom there are no child subnetworks, or for whom all of the child subnetworks are leaves and none of them have members of the

destination group, that router generates a *prune message* for the particular (source, group) pair and sends it back to the router that is one hop towards the source. If the one-hop-back router receives prune messages from all of its subordinate routers (i.e., routers attached to its child subnetworks that use those subnetworks to reach the source), and if its child subnetworks also have no members of the destination group, it in turn sends a prune message back to its predecessor. In this way, information about the absence of group members propagates back up the tree towards the source, along all branches that do not lead to group members. Subsequent packets from the same source to the same group are blocked from traveling down the unnecessary branches by the routers at the heads of those branches.

In order that routers do not have to remember pruning information about a particular (source, group) pair forever, prune messages have an associated *lifetime*, which limits how long the receiver is expected to remember receiving it. A prune message from a router that has no subordinate routers is sent with a lifetime $T_L$ which is configurable by the internetwork administrators; a prune messages from a router that has subordinates is sent with a lifetime that is the minimum of the remaining lifetimes of those prune messages received from its subordinates. Thus, any branch that is pruned from the tree "grows back" within an interval of $T_L$. If, at that time, there is still traffic from the same source to the same group, the next multicast packet triggers the generation of new prune messages, as long as there are still no members of the group on that branch. If there is no further traffic, the routers are left with no memory of that particular (source, group) pair.

More details are shown in the pseudo-code of Figure 7.5.

This code replaces corresponding parts of the DV truncated-broadcast algorithm of Figure 7.3. (So, for example, event 1 is not repeated in this figure, since it is unchanged.) The expressions enclosed in <> brackets are invocations of the procedures listed in Figure 7.6.

The algorithm operates on the same types of objects as the DV truncated-broadcast algorithm, with the following exceptions: One additional attribute already present in the DV routing table entries is used:

$r$.next-hop-address address of next hop towards destination

```
2: on change of information about destination d from subnetwork k:
      if ∃ route r such that r.destination = d
            ...
            r.num-subordinates[k] ← number of neighbors n such that
                                    n.subnet = k and n.distance[d] = infinity

3: on receipt of multicast packet from source s to group g via subnetwork k:
      if ∃ route r such that r.destination = s and r.next-hop-subnet = k
      and ∃ no prune-sent ps such that ps.source = s and ps.group = g
            copies ← 0
            for each subnetwork j
                  if <should forward from s to g on j, given r>
                        send copy of packet on subnetwork j
                        copies ← copies + 1
            if copies = 0
                  <generate prune message for s, g, given r>

4: on receipt of prune-message(s, g, t) from address a:
      if ∃ neighbor n such that n.address = a and n.distance[s] = infinity
      and ∃ route r such that r.destination = s and r.child[n.subnet] = true
            <record prune message from neighbor n for s, g, t>
            copies ← 0
            for each subnetwork j
                  if <should forward from s to g on j, given r>
                        copies ← copies + 1
            if copies = 0
                  <generate prune message for s,g, given r>

5: on expiration of ps.lifetime for prune-sent ps:
      delete ps

6: on expiration of pr.lifetime for prune-received pr:
      delete pr
```

Figure 7.5: DV Multicast Algorithm (Partial)

```
<should forward from source s to group g on subnetwork j, given route r>:
        r.child[j] and
        ((r.num-subordinates[j] > number of prunes-received pr such that
                pr.source = s and pr.group = g and pr.subnet = j) or
        (∃ group presence p such that p.group = g and p.subnet = j))

<generate prune message for source s, group g, given route r>:
        if ∃ prune-sent ps such that ps.source = s and ps.group = g
                ps.timeleft ← T_L
        else
                create prune-sent ps
                ps.source ← s; ps.group ← g; ps.timeleft ← T_L
        for each prune-received pr such that pr.source = s and pr.group = g
                if pr.timeleft < ps.timeleft
                        ps.timeleft = pr.timeleft
        if s ≠ r.next-hop-address
                send prune-message(s, g, ps.timeleft) to r.next-hop-address

<record prune message from neighbor n for source s, group g, lifetime t>:
        t ← t − k.transit-time, where k = n.subnet
        if ∃ prune-received pr such that
        pr.source = s and pr.group = g and pr.address = n.address
                pr.timeleft ← t
        else
                create prune-received pr
                pr.source ← s; pr.group ← g; pr.timeleft ← t
                pr.address ← n.address; pr.subnet ← n.subnet
```

Figure 7.6: DV Multicast Algorithm—Supplemental Procedures

and the $r$.leaf attribute is replaced by:

$r$.num-subordinates[$k$] # of subordinates, per attached subnetwork $k$

To the "attached subnetwork" object, one more attribute is added:

$k$.transit-time maximum time to send packet across subnetwork $k$

Two additional types of objects are maintained. A "prune-received" object $pr$ represents memory of a prune message received from a subordinate router. It has the following attributes:

| | |
|---|---|
| $pr$.source | source of multicast packets |
| $pr$.group | destination of multicast packets |
| $pr$.address | address of subordinate router |
| $pr$.subnet | subnetwork of subordinate router |
| $pr$.lifetime | remaining lifetime for received prune message |

A "prune-sent" object $ps$ represents memory of a prune message sent to a previous-hop router. It has the following attributes:

| | |
|---|---|
| $ps$.source | source of multicast packets |
| $ps$.group | destination of multicast packets |
| $ps$.lifetime | remaining lifetime for sent prune message |

The expression "prune-message($s$, $g$, $t$)" means a prune message for multicasts from source $s$ to group $g$, with a lifetime of $t$.

The assignment to $r$.num-subordinates[$k$] under event 2 replaces the corresponding assignment to $r$.leaf[$k$] in the truncated-broadcast algorithm. Whereas the previous algorithm was only concerned with the presence or absence of a subordinate router on a subnetwork, this algorithm must keep track of how many subordinates are present.

The pseudo-code under event 3 shows the revised forwarding algorithm for multicast delivery. In order to be considered for forwarding, a multicast packet must meet two conditions: (1) it must arrive via the subnet used to reach the source of the packet (this is the normal RPF requirement), and (2) there must be no record of a prune message having been sent for the packet's source and group. The latter

condition is an optimization to avoid the overhead of making forwarding decisions when the router has previously discovered that it does not have to forward. If the two conditions are satisfied, the router determines, for each attached subnetwork, whether or not to forward a copy of the multicast packet. This determination is made as shown by the pseudo-code for the predicate "<should forward ...>": a copy should be forwarded to any child subnetwork on which either there is at least one subordinate router that has not sent a prune message for the packet's source and group, or there are members of the packet's destination group. While making this determination for each subnetwork, the router keeps track of how many copies are actually sent (in the local variable "copies"); if none are sent, then a prune message is generated for the previous-hop router. The pseudo-code to generate the prune message computes an appropriate lifetime, based on the lifetimes of prune messages received from subordinates, if any. It also handles the special case of the previous hop being the multicast source itself, rather than another router; in that case, no prune message is sent, because the source is not expected to participate in the routing protocol (however, the router creates a record of having sent a prune message, in order to speed up the recognition and discarding of later packets from the same source).

A router may continue to receive multicast packets for a given (source, group) pair from the "parent" subnetwork, after sending a prune message for that source and group, for three reasons:

- The source may be directly attached to that subnetwork.
- There may be members of the destination group on the subnetwork, or other subordinate routers of that subnetwork that have not generated prune messages for the (source, group) pair.
- The router's prune message may have been lost in the subnetwork.

The last circumstance could be remedied by requiring prune messages to be acknowledged by the receiving router. However, given the low probability of packet loss in modern subnetworks, and the possibility that either of the other two circumstances might also pertain, it is not worth the extra complexity and bandwidth cost of performing prune acknowledgements and retransmissions on such subnetworks. If the

unwanted multicast traffic continues for longer than the lifetime of the lost prune message, another prune message will then be generated.

The pseudo-code for event 4 shows the router behavior on reception of a prune message from a subordinate router. If the message passes some validity checks, a record is made of the received message. When recording the lifetime, the maximum time that the message might have been in transit is taken into account; it is harmless if the prune record expires a little earlier than necessary, and that is preferable to expiring late (a pruned branch should "grow" from the root towards the leaves), so the estimate of the maximum transit time should err on the side of being too long rather than too short. After the received prune message is recorded, the router determines whether or not it should in turn send a prune message to its predecessor, using the same decision procedure as used when forwarding multicast packets.

Records of sent and received prune messages are discarded when their lifetimes expire (events 5 and 6).

The algorithm as described so far is not sufficiently responsive either to the appearance of group members on new subnetworks or to a change of topology, when either of those events affects an active multicast tree—adaptation to those events may not occur until the lifetimes of some prune records expire, which compromises our goal of low join latency. To remedy this problem, we add a mechanism for quickly "grafting" a pruned branch back onto a multicast tree. The pseudo-code for this mechanism is quite complicated, so we have left it out of the figure, but it is simple to explain: a router can cancel a previously sent prune message by sending a *graft message* to the same router; the graft message is propagated as far as necessary to rejoin the originating router into the specified multicast tree. The circumstances under which a router generates a graft message for source $s$ and group $g$, within the lifetime of a previously sent prune message for $s$ and $g$, are the following:

- Group $g$ appears on a child subnetwork for $s$.
- A subnetwork on which $g$ is present becomes a new child for $s$.
- A new subordinate router for $s$ appears on a child subnetwork.
- A graft message for $s$ and $g$ is received from a subordinate router for $s$.

When such events occur, they may require grafting onto multiple active multicast trees. For example, the appearance of a new group on a subnetwork affects the trees of all sources sending to that group for which the subnetwork is a child. Therefore, graft messages can carry a list of sources or a list of groups, in order to request cancellation of all previously sent prune messages referring to those sources or those groups.

Unlike prune messages, it is important that graft messages be reliably delivered. Loss of a graft message can cause a branch to be involuntarily omitted from a multicast tree for up to the maximum prune lifetime, $T_L$. Therefore, a graft message is acknowledged by the receiver, and retransmitted by the sender until the acknowledgement is received.

Finally, there are some circumstances, other than reception of a graft message, in which a router must discard prune records before their lifetime has expired. Records of received prune messages are discarded when the subordinate routers that generated them cease to be subordinates (i.e., they fail, or they stop reporting a distance of infinity, or the subnetworks to which they are attached cease to be child subnetworks). Records of transmitted prune messages are discarded when the router to which they were sent ceases to be the next-hop router for the sources identified in them.

### 7.3.2 Costs of the DV Multicast Algorithm

The storage costs of the DV multicast routing algorithm are of two types: additional fields in the routing table and space to record transmitted and received prune messages. The additions to the routing table are the child and num-subordinates fields in each route entry. The child field can be implemented as a bit map, with one bit for each attached subnetwork, the same as for truncated-broadcast. The num-subordinates field may be implemented as an array of small integers, one integer per attached subnetwork. A router supporting 8 attached subnetworks, with up to 16 subordinate routers per subnetwork, would therefore require 8 bits for a child field and and 32 bits (8 subnetworks times 4 bits per subnetwork) for a num-subordinates field, for a total of 40 bits or 5 bytes in every route entry (as compared to 2 bytes per route entry for the truncated-broadcast algorithm). This would add 30% to the size

of a routing table whose normal route entries are 16 bytes, the minimum reasonable size for DV routing (assuming 4-byte DARPA IP addresses); many implementations of DV routing have route entries much larger than 16 bytes[5], for which the addition of 5 more bytes would be correspondingly less significant.

Potentially more significant is the space required to record transmitted and received prune messages. Each prune message sent or received by a router requires approximately 20 bytes of storage in that router for the lifetime of the message, assuming the use of 4-byte DARPA IP addresses. The number of such records that must be stored in a router depends on the number of active multicast trees in the internetwork and the position of the router relative to those trees. An interesting property of this algorithm is that those routers that are interior to an active multicast tree do not store any information about that tree—it is all the routers that are outside of the multicast tree or on the periphery of the tree that must store pruning information about the tree.

The number of prune records that must be stored by a router $r$, per active tree, is no more than the number of $r$'s subordinate routers in the (unpruned) tree, plus one for the previous-hop router in the tree. That number is bounded by the number of neighboring routers of $r$, and is often much less than that bound. The number of active trees depends on the choice of maximum prune lifetime, $T_L$: an active tree is one on which at least one multicast packet has been sent within the last $T_L$ interval. The larger $T_L$, the greater number of active trees are likely to exist and the more storage is required for prune records.[6] Unfortunately, there is currently no data on which to base a choice of $T_L$. An important task for future research, once internetwork multicast applications are deployed, is to measure the "locality" or "working set size" of multicast activity, that is, the number of active trees out of all potential trees, over various time intervals.

The choice of $T_L$ controls a trade-off between storage and bandwidth: with a

[5]For example, 4.3BSD UNIX uses 128-byte route entries.

[6]Recall that a tree is defined by a (source, group) pair, and that a source need not be a single host. For example, in an internetwork based on DARPA IP, in which addresses contain an embedded subnetwork number, all multicast packets originating on the same subnetwork, sent to the same multicast group, are treated as belonging to a single tree, regardless of the number of sending hosts.

$T_L$ of zero (i.e., no pruning), no storage is required and all multicast packets are delivered by truncated-broadcast; with a $T_L$ of infinity, storage is required for every tree that has ever been used and all multicast packets except the first on a tree are delivered by pruned multicast. At intermediate values of $T_L$, storage is required only for active trees, and every $T_L$ interval a multicast packet from each active tree is delivered by truncated-broadcast. The truncated-broadcast triggers prune messages on those subnetworks that are traversed unnecessarily; the number of such messages on each subnetwork is less than or equal to the number of subordinate routers on that subnetwork. Thus on each subnetwork, a small, bounded amount of bandwidth is spent to avoid the cost of any unnecessary multicast packet transmissions for an interval of $T_L$, for each active tree.

The storage/bandwidth trade-off offers routers a graceful way to handle temporary storage exhaustion: if a router runs out of space to store prune records, it may simply ignore incoming prune messages and refrain from sending prune messages, allowing some multicast trees to be less-well pruned until storage again becomes available.

Another bandwidth cost of the algorithm is the cost of sending graft messages and graft acknowledgements. These messages are caused by the appearance of groups on new subnetworks and by topology changes; whether or not such an event actually incurs any graft messages depends on its specific effect on multicast trees that are active at the time of the event. In a stable, operational internetwork, topology changes should be infrequent enough that the number of graft messages due to topology changes will be insignificant. Grafting due to group appearance, however, might be common, for example, if there are applications that constantly multicast information to a frequently changing population. (An example might be a weather report service.) The "hysteresis" in the Host Membership Protocol (that is, the delay between the time a group is reported present on a subnetwork and the earliest time it can be reported absent) provides some limit on the rate of prune and graft message generation as a result of frequent changes of group membership on any particular subnetwork. If necessary, limits may also be imposed on the rate of graft-message generation by any given router, trading off an occasional increase in join latency for reduced bandwidth consumption by graft messages and their acknowledgements.

The algorithm incurs processing costs in the generation, reception, and interpretation of prune and graft messages; those costs are proportional to the number of such messages that must be handled, which is discussed above. The algorithm as presented in pseudo-code also involves significant processing when making the routing decision for each received multicast packet: the router must retrieve and examine one route entry, some number of group presence records, and some number of prune records. This processing burden can be significantly reduced by caching the result of the routing decision for a given (source, group) pair (perhaps in the form of a bit map identifying those attached subnetworks that are to receive copies of multicasts with that source and group), and using that cached information for forwarding subsequent packets for the same source and group. This would require that even those routers in the interior of an active multicast tree keep a small amount of information about that tree.

Finally, it should be noted that, since prune messages are triggered by reception of multicast packets, any packets sent with small hop limits incur no pruning or caching costs in subnetworks and routers beyond the reach of those packets. Thus, applications that can exploit hop-limited multicasts, such as those that perform bounded, expanding-ring searches as discussed in Section 3.3, do not contribute to the total cost of supporting multicasting as the internetwork grows.

## 7.4 Chapter Summary

In this chapter, we have described truncated-broadcast and multicast extensions to distance-vector routing, based on refinements of Dalal and Metcalfe's reverse-path forwarding algorithm. As long as the the routers assign the same length to both directions of a path (as when using hop count as the distance measure, which is usually the case for distance-vector routers), our algorithms deliver multicasts along paths as good as those used for unicast packets to the same destinations, thus satisfying the reliability and performance requirements of the Host Group Model.

Both the truncated-broadcast and multicast algorithms require a modest increase (typically, less than 6 bytes) to the size of the routing table entries maintained by

a distance-vector router, and a corresponding processing cost to maintain the extra information; those costs grow at the same rate as the unicast routing costs, thus imposing no new limit on the scalability of the internetwork. In order to eliminate most of the excess packet hops of truncated-broadcast delivery, the multicast algorithm incurs additional overhead to learn and cache information about currently-active multicast senders and their destination groups. Those costs are, to a first approximation, proportional to the number of active (source, group) pairs in the internetwork. However, as the internetwork grows, the number of groups to which any source is actively sending is expected to grow much more slowly than the number of sources, so the costs of the multicast algorithm are expected to scale almost linearly with the number of active multicast sources. That is as good as as, or better than, the costs of the underlying unicast algorithm, which scale linearly with the *total* number of sources, including those that are not actively sending multicasts.

# Chapter 8

# Link-State Multicast Routing

The third major routing style to be considered is that of link-state routing, also known as "New Arpanet" or "Shortest-Path-First" (SPF) routing [50]. As well as being used in the Arpanet, the link-state algorithm has been proposed by ANSI as an ISO standard for intra-domain routing [45], and is being considered as a standard "open" routing protocol for the DARPA Internet [54].

## 8.1 Overview of Link-State (LS) Routing

Under the link-state (LS) routing algorithm, every router monitors the state of each of its incident links or subnetworks. The state of a subnetwork includes whether or not the subnetwork (or the router's attachment to the subnetwork) is operational, the set of neighboring routers reachable across the subnetwork, and in some cases, a measure of the traffic load on the subnetwork. Whenever the state of a subnetwork changes, the routers attached to that subnetwork broadcast the new state to every other router in the internetwork. The broadcast is accomplished by a special-purpose, high-priority flooding protocol that ensures that every router quickly and reliably learns the new state. Consequently, all routers know the state of all subnetworks and all routers, from which they can each determine the complete topology of the internetwork. Given the complete topology, each router independently computes the shortest paths from itself to every possible destination, using Dijkstra's algorithm

[3]. From those paths, it records the first-hop subnetwork and first-hop router in a routing table entry for each destination, to be used for forwarding packets.

In order to reduce the broadcast traffic resulting from a state change, some versions of the LS protocol [45, 54] elect a single *designated router* on each subnetwork to report state changes for that subnetwork. For the purpose of our multicast routing extensions, we assume that this feature is provided.

## 8.2 LS Truncated-Broadcast Routing

Given the complete topological knowledge available in every LS router, any router can directly compute the shortest-path broadcast tree rooted at any particular source $s$, using Dijkstra's algorithm. From its own position in that tree, the router can determine its parent subnetwork, from which it should receive broadcasts from $s$, and any child subnetworks, onto which it must forward broadcasts from $s$. The router can also identify, from the tree computation, which of its child subnetworks are leaf subnetworks in the broadcast tree for $s$; this, along with the Host Membership Protocol, provides the information necessary to perform truncated-broadcast routing of multicast packets.

### 8.2.1 Description of the LS Truncated-Broadcast Algorithm

Our LS truncated-broadcast algorithm is shown in Figure 8.1. It is used in conjunction with a link-state routing algorithm and with the router part of the Host Membership Protocol. The algorithm operates on "route" objects, which are the entries in the LS routing table; a route object $r$ has the following attributes of relevance to this algorithm:

| | |
|---|---|
| $r$.destination | internetwork destination address |
| $r$.parent | parent subnetwork, for broadcasts from destination |
| $r$.child[$k$] | child flags, per attached subnetwork $k$ |
| $r$.leaf[$k$] | leaf flags, per attached subnetwork $k$ |

```
1: on change of designated router on subnetwork k:
       k.interrogator = k.designated

2: on change of topology:
       for each route r
              r.parent = undefined

3: on receipt of multicast packet from source s to group g via subnetwork k:
       if ∃ route r such that r.destination = s
              if r.parent = undefined
                     <compute r.parent, r.child, r.leaf attributes>
              if r.parent = k
                     for each subnetwork j
                            if r.child[j] and
                            ((not r.leaf[j]) or (∃ group presence p such that
                            p.group = g and p.subnet = j))
                                   send copy of packet on subnetwork j
```

Figure 8.1: LS Truncated-Broadcast Algorithm

The first field is normally present in an LS routing entry; the other three are added for the purposes of this algorithm.

The algorithm also operates on the "group presence" and "attached subnetwork" objects as defined for the router part of the Host Membership Protocol (page 47), except that each attached subnetwork object $k$ has one additional attribute:

$k$.designated true if this router is LS designated router for subnetwork $k$

The election of a designated router by the LS algorithm (event 1) serves as an election of a membership interrogator, as required by the Host Membership Protocol.

On any change of topology, the normal LS routing algorithm recomputes its entire routing table. As shown in the pseudo-code for event 2, the parent field in each routing entry is initialized at that point to a special value meaning "undefined".

The truncated-broadcast delivery of multicast packets is performed as shown in the pseudo-code for event 3. The source of an incoming multicast packet is looked up in the routing table. If the route entry contains an undefined parent value, that indicates that this is the first multicast packet from that particular source since the last topology change; in that case, the parent, child, and leaf fields are computed, as described in the next paragraph. The router then checks that the packet arrived on the correct parent subnetwork for its source and, if so, forwards it on all child networks that are not leaf subnetworks or that have members of the destination group, which is the normal truncated-broadcast forwarding procedure.[1]

The determination of the parent, child, and leaf values for a route entry entails the execution of Dijkstra's algorithm to compute the shortest-path broadcast tree rooted at the source identified in the route entry. That algorithm works by starting with a node for the root, and iteratively adding all other nodes to the tree, nearest nodes first. There is a node for every router and every subnetwork in the internetwork; the information about which nodes are directly attached to which others is provided

[1]The only difference between this forwarding procedure and that of our DV truncated-broadcast algorithm, shown under event 3 in Figure 7.3, is in the determination of the "correct" arrival subnetwork: in the DV case, the arrival subnetwork must be the next-hop subnetwork on the shortest path back to the source of the packet, whereas in the LS case, the arrival subnetwork must be the parent subnetwork computed from the shortest-path tree rooted at the source. This is the difference between reverse-path forwarding and forward-path forwarding.

by the underlying LS routing protocol. When the router that is performing the computation adds the node for itself to the tree, it records as the parent subnetwork that subnetwork to which its node is added. (The nodes in a path alternate between router nodes and subnetwork nodes.) When the router adds a descendent subnetwork node to its own node, it sets the child and leaf flags for that subnetwork. When the router adds a descendent router node to one of those child subnetwork nodes, it clears the leaf flag for that subnetwork. When all the nodes have been added to the tree, the child and leaf information is complete.

### 8.2.2 Costs of the LS Truncated-Broadcast Algorithm

The LS truncated broadcast algorithm incurs the storage cost of the parent, child, and link fields added to the routing table entries. The parent field is a small integer identifying an attached subnetwork, and the child and leaf fields are bit maps, each with one bit per attached subnetwork. Three bytes per route entry would be sufficient to hold these three fields, in a router supporting up to 8 attached subnetworks. This is an insignificant cost for current-generation routers.

The most significant cost of the algorithm is the cost of computing the broadcast trees for each multicast source. Figure 8.2 shows the time needed (worst-case, across all possible sources) to compute a single shortest-path tree by Dijkstra's algorithm in some internetworks of various sizes[2], using a contemporary, RISC-based processor[3]. The lower graph is a magnification of the low end of the upper graph. The two curves correspond to whether or not the metric for measuring path length is constrained to a small range of discrete values, or can take on an arbitrary value.[4] As can be seen from the graphs, the computation time (regardless of the type of metric) is a small number

[2] The one with approximately 70 subnetworks is the Stanford University campus internetwork as of August 1988; it is composed of 71 Ethernets, haphazardly connected by 48 routers. All of the other internetworks were synthesized from that one: smaller ones by partitioning, and larger ones by replication and interconnection.

[3] DECstation 3100.

[4] The link-state routing protocol proposed by ANSI for intra-domain routing of ISO connectionless network traffic [45] specifies a constrained range of values for path metrics, in order to improve the scalability of the tree computation. The improvement comes from replacing an $O(\log n)$ sort step in Dijkstra's algorithm with an O(1) bucket sort.

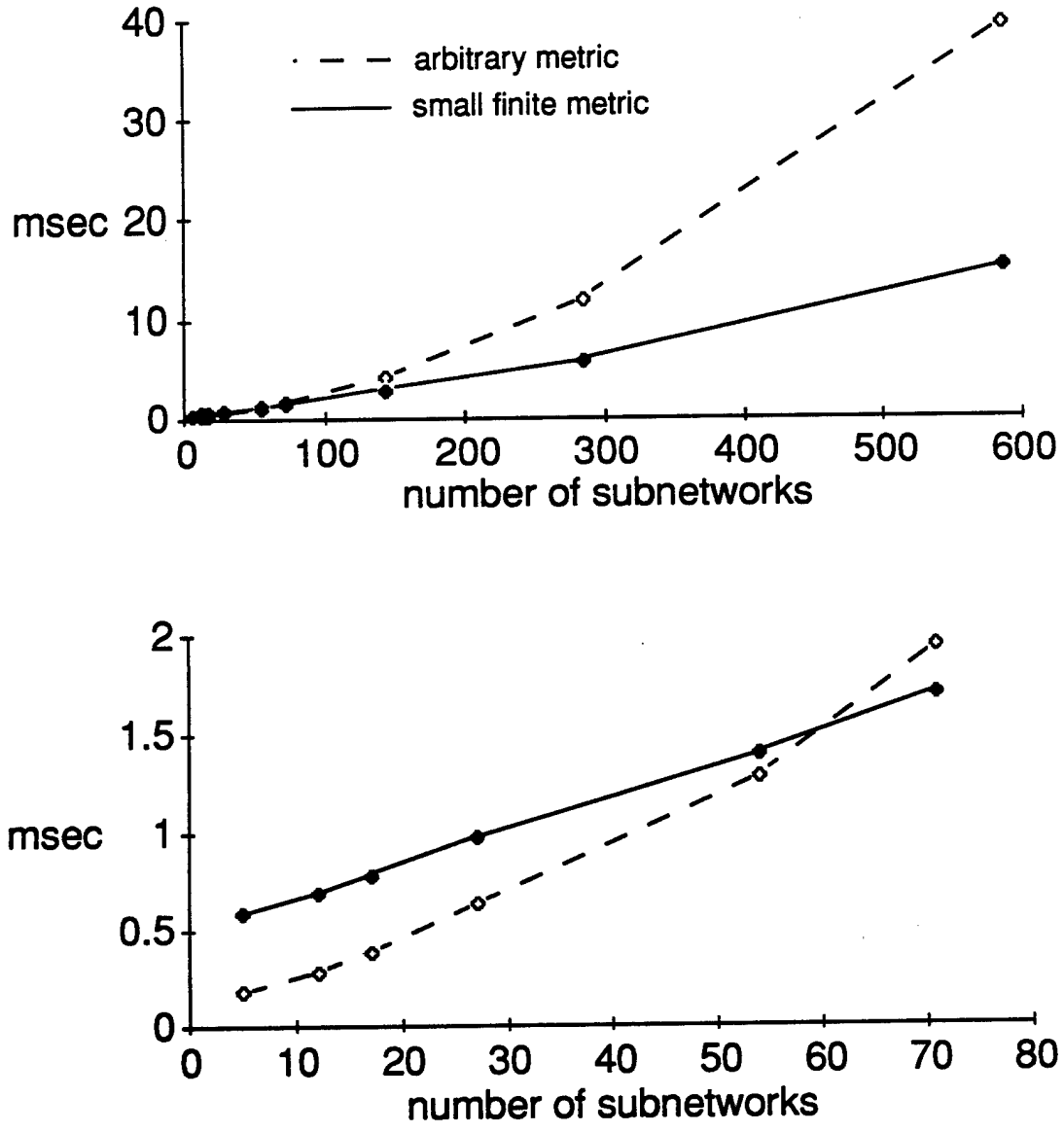



Figure 8.2: Shortest-Path Tree Computation Time (Dijkstra's Algorithm)

of milliseconds for internetworks with fewer than a hundred or so subnetworks, which is the case for most single-level internetworks—when internetworks grow beyond that size, they are usually divided into multiple routing domains to simplify administration or to avoid other scaling problems, such as routing traffic overhead. Apart from the fact that the speed of processors available for this application continues to increase, there are a couple of factors that further suggest that the computational demands of the algorithm are reasonable:

- Unlike the tree computation that is performed to regenerate the routing table on every topology change, the broadcast tree computations need not hold up the forwarding of other traffic—they may be performed in parallel in a multiprocessor-based router, or performed as a lower-priority task in a multi-threaded uniprocessor implementation.

- By delaying the computation of each broadcast tree until triggered by a multicast packet, rather than computing all trees on every topology change, the computational load is spread over a longer period of time, and some computation is avoided when not all possible sources send multicast packets between successive topology changes (or when all the multicast packets from some sources have hop limits too small to reach some routers).

The original link-state algorithm developed for the Arpanet [50] includes a method for avoiding a full tree computation on every topology change, exploiting the fact that most topology changes affect only part of a previously-computed tree. However, it requires that the entire tree be retained for examination when the next topology change occurs. Our algorithm discards a tree image after the parent, child, and leaf information has been extracted from it; retaining a full tree for every active multicast source would significantly increase the storage requirement of the algorithm. Furthermore, the method for avoiding the full computation has been dropped from more recent versions of the link-state protocol, because its computational cost is often not much less than the cost of a full tree computation [45].

The other cost of this algorithm is the bandwidth cost of forwarding multicast packets further than necessary, inherent in the truncated-broadcast delivery strategy.

The LS multicast algorithm described next addresses that problem.

## 8.3 LS Multicasting Routing

Our final multicast routing algorithm for single-level internetworks is the link-state multicast algorithm, which is an extension of our LS truncated-broadcast algorithm. The main idea behind this algorithm is to consider the set of groups present on a subnetwork to be part of the state of that subnetwork—whenever the set of groups changes, as with any other state change, the designated router for that subnetwork floods a link-state update message to all other routers, reporting the change. The existing flooding protocol ensures that all routers end up with complete and consistent knowledge of which groups are present on which subnetworks. Adding that information to the topology information already provided to every router allows any router to directly compute the multicast delivery tree from any source to any group.

### 8.3.1 Description of the LS Multicast Algorithm

Our LS multicast algorithm is shown in Figure 8.3. It is used in conjunction with a link-state routing algorithm and with the router part of the Host Membership Protocol. The algorithm operates on "cache record" objects that hold the information required to forward multicast packets; a cache record $c$ has the following attributes:

| | |
|---|---|
| $c$.source | source of multicast packets |
| $c$.group | destination of multicast packets |
| $c$.parent | parent subnetwork, for multicasts from $c$.source |
| $c$.min-hops[$k$] | min. hops to $c$.group, per attached subnetwork $k$ |

The algorithm also operates on "attached subnetwork" objects as defined for the router part of the Host Membership Protocol (page 47), except that each attached subnetwork object $k$ has one additional attribute:

| | |
|---|---|
| $k$.designated | true if this router is LS designated router for subnetwork $k$ |

```
1: on change of designated router on subnetwork k:
       k.interrogator = k.designated

2: on appearance or disappearance of group g on subnetwork k:
       if k.designated
              flood link-state update, reporting change of g on k

3: on generation or receipt of link-state update, reporting change of g:
       for each cache record c such that c.group = g
              discard c

4: on change of topology:
       for each cache record c
              discard c

5: on receipt of multicast packet from source s to group g via subnetwork k,
       with remaining hop limit hl:
       if ∃ no cache record c such that c.source = s and c.group = g
              create cache record c; c.source = s; c.group = g
              <compute c.parent and c.min-hops attributes>
       if c.parent = k
              hl = hl − 1
              for each subnetwork j
                     if hl ≥ min-hops[j]
                            send copy of packet on subnetwork j
```

Figure 8.3: LS Multicast Algorithm

The election of a designated router by the LS algorithm (event 1) serves as an election of a membership interrogator, as required by the Host Membership Protocol.

Whenever the procedures for the Host Membership Protocol detect the appearance of a new group on an attached subnetwork or the disappearance of a previously-present group (event 2), if the router is the LS designated router for that subnetwork, it invokes the LS protocol procedures to flood a link-state update, reporting the change.

When a link-state update for a group is generated by the procedure for event 2 or received from another router (event 3), all cache records referring to that group are discarded, since the change may have invalidated them. Similarly, whenever there is a topology change (event 4), *all* cache records are discarded.

The multicast forwarding procedure for packet from source $s$ to group $g$ (event 5) is as follows: Look up the cache record for that (source, group) pair; if there is no such record, create one and compute its parent and min-hops attributes as described in the next paragraph. (If the cache storage is exhausted, the least-recently-used cache record is discarded to make room.) If the existing or newly-created cache record shows that the packet arrived via the correct parent subnetwork, decrement the packet's hop-limit and proceed to forward a copy on every subnetwork via which, according to the cache record, a member of the destination group is reachable within the remaining hop-limit in the packet.

The computation of the parent and min-hops attributes for a new cache record is similar to that described for LS truncated-broadcast. Dijkstra's algorithm is used to compute the shortest-path broadcast tree rooted at the multicast source. As the node for this router (i.e., the router performing the computation) is added to the tree, its parent subnetwork is recorded, and each element of the min-hops array is initialized to a value meaning "infinity". As each descendent subnetwork node is added to the tree, its distance in hops from this router is computed, based on the hop distance of its ancestor. When the first descendent subnetwork node which is reachable via this router's subnetwork $k$, and which has members of the destination group, is added to the tree, its hop distance is recorded in the $k$th element of the min-hops array. When all of the nodes have been added to the tree, the min-hops array contains non-infinite

values for all subnetworks that lead from this router to members of the destination group, in the tree rooted at the source; those values indicate the distance in hops to the nearest members.

### 8.3.2 Costs of the LS Multicast Algorithm

The costs of the LS multicast algorithm may be divided into two categories: costs related to forwarding multicast packets and costs related to disseminating and maintaining group membership information.

In support of multicast forwarding, the algorithm incurs the storage cost of the multicast cache records and the tree computation costs of cache misses, each of which may be traded off in favor of the other. Each cache record requires approximately 20 bytes of storage, assuming the use of DARPA IP addresses, so it is reasonable to cache thousands of records. A router caches information about only those multicast trees that pass through, or terminate at, that router. The cost of each tree computation is the same as for the LS truncated-broadcast algorithm, discussed previously.

The dissemination and maintenance of group membership information incurs the cost in each router of storing the list of groups present on each subnetwork, and the bandwidth cost of the link-state updates triggered by group membership changes. Assuming that there are generally fewer groups present on a single subnetwork than there are individual hosts, the storage required in the routers is less than that needed to record all host addresses, as is necessary in some versions of the LS algorithm [45]. On the other hand, the frequency of appearance and disappearance of groups on a subnetwork would normally be higher than that of hosts, resulting in more link-state update traffic. (The reliable flooding protocol used to disseminate an update requires every router to send a packet—either an update or an acknowledgement—on each of its subnetworks, except those subnetworks that have no other routers attached.) Nonetheless, based on the following two observations, we do not expect the bandwidth cost of such traffic to be significant:

- Most memberships are long-lived. For well-known service groups, each member host joins the group when it starts up and remains a member until it fails or is

restarted. Some transient groups, such as those supporting human conferencing, are also relatively long-lived. Such groups are not expected to be the source of rapid join/leave updates.

- Volatile groups, i.e., groups with short-lived memberships, tend to be sparsely distributed. An example is a group set up for a short, distributed computation. Such groups generate rapid join/leave updates from only a small number of subnetworks at a time.

Furthermore, the routers may easily bound the overhead of group membership updates, protecting themselves and the subnetworks from unusual bursts of membership activity and from misbehaving group members, as follows:

- The link-state updates generated when groups disappear from a link may safely be delayed and piggybacked on other updates. The only consequence of delaying a report of group disappearance is the possible forwarding of packets onto subnetworks where there are no longer any destination group members, until the update is eventually sent.

- If necessary, updates generated when a group first appears on a subnetwork may be rate-limited, at the cost of greater join latency. For example, a router may be prohibited from generating group appearance updates more frequently than once every five seconds, with each update listing all groups that appeared in the past five seconds. This scheme is already used to limit the rate of normal link-state updates caused by a subnetwork that is oscillating between being in the "up" and "down" states.

As an example, if a once-per-five-second update limit were imposed on the 71-subnetwork topology referred to in footnote 2, page 105, in the unlikely (worst) case that every subnetwork acquired at least one new group every five seconds, it would result in 48 update messages per second on each of 24 transit subnetworks[5]. Assuming an update message size of 100 bytes, those messages would consume only 4% of the bandwidth of those Ethernets.

[5]i.e., subnetworks with more than one attached router.

## 8.4 Chapter Summary

In this chapter, we have described truncated-broadcast and multicast extensions to the link-state routing algorithm. Like the algorithms in the preceding two chapters, these link-state algorithms satisfy the reliability and performance requirements of the Host Group Model, by ensuring that multicast packets are delivered along the same (or equally good) paths as unicast packets to the same destinations.

The costs of these algorithms are modest, relative to the available storage, processing, and bandwidth resources in modern internetworks. Both algorithms incur per-router costs that scale linearly with the number of subnetworks, the same as the costs of the underlying unicast routing algorithm. In addition, the multicast version incurs the cost of caching information about active multicast sources and their destination groups, in order to eliminate the the excess packet hops of the truncated-broadcast version. Since the average number of destination groups per active source is unlikely to grow significantly as the internetwork grows (i.e., as the number of active sources grows), the caching costs are expected to scale almost linearly with the number of active multicast sources, which is no worse than the scaling of the underlying unicast algorithm.

# Chapter 9

# Hierarchical Multicast Routing

Multicast routing can be scaled up to handle very large internetworks in the same way as unicast routing—by using *hierarchy*. Under a hierarchical routing scheme [47], an internetwork is divided into non-overlapping *regions*, each containing some number of subnetworks and routers. The routers inside each region run a routing algorithm for routing within that region. The routers that interconnect regions run a separate algorithm (or another instance of the same algorithm) for routing between regions, in essence, treating each region as if it were a single, virtual subnetwork, as illustrated in Figure 9.1. The same approach can be applied recursively, dividing regions into sub-regions and so on. The benefit of hierarchical routing is that it allows the number of subnetworks (or virtual subnetworks) that must be known to any single router to be bounded. Since the per-router costs of most unicast routing algorithms, and all of our multicast routing algorithms, are, to first order, proportional to the number of subnetworks, hierarchical structuring eliminates the dominant scaling limit of those algorithms.

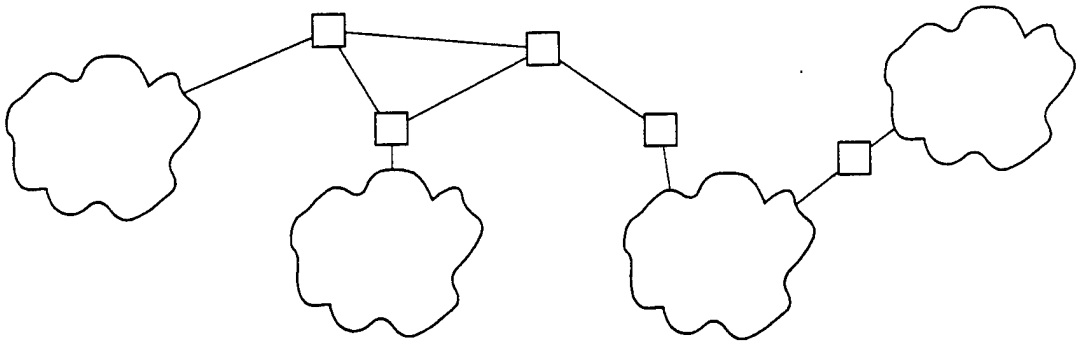


Figure 9.1: A Hierarchical Internetwork

## 9.1 Using the Multicast Algorithms Hierarchically

In order for a region to play the role of a single subnetwork, from the point of view of a multicast routing algorithm running in the inter-region ("backbone") routers, the region must satisfy certain assumptions of that algorithm. In particular, all of our algorithms assume that a multicast packet sent into a subnetwork will be delivered to all destination group members attached to that subnetwork, *plus* all attached routers; if the subnetwork is really a region, "all attached routers" means all backbone routers connected to that region. Any of the multicast algorithms we have described can be used within the region to accomplish delivery to all internal group members, and they may easily be modified to deliver copies to any attached backbone routers, as follows:

- The ST or LS multicast algorithms may be modified to recognize a special "wildcard" group that each backbone router can join. A copy of any multicast packet, regardless of destination address, is sent to all wildcard members. Information about wildcard memberships is propagated by the same mechanisms as normal memberships.

- For any of the truncated-broadcast algorithms, each backbone router can ensure that the (physical) subnetwork to which it is attached never appears to be a leaf subnetwork in any multicast tree (for example, by pretending to be attached to another, "phantom" subnetwork, one hop beyond the attachment subnetwork).

That will cause all multicast packets to appear on the attachment subnetwork and, thus, at the backbone router.

- For the DV multicast algorithm, the same method as for the truncated-broadcast algorithms can be used, with an additional modification to ensure that a backbone router never sends any prune messages into the region.

The requirement that all attached backbone routers receive a copy of any intra-region multicasts can be relaxed in the special, but common, case of *stub regions*, that is, regions that are not used for transit traffic between other regions. Many campus or single-site internetworks are stub regions of a larger, wide-area internetwork. Also, some hierarchical (unicast) routing protocols, such as ANSI's IS-IS [45], require that all bottom level regions ("level 1 areas", in IS-IS terminology) be stub regions. In those cases, it is sufficient to deliver a copy of all intra-region multicasts to only *one* of the backbone routers. (Since the region is not used for transit traffic, the backbone routers can be assumed to have connectivity to each other outside of the region.) The backbone routers are generally able to detect that a region is a stub region, and can choose among themselves which one will join the wildcard group in the region or otherwise arrange to receive all intra-region multicasts.

Our ST and LS multicast algorithms, when used for inter-region routing, place another requirement on the intra-region algorithms: the backbone routers must be able to detect the appearance and disappearance of groups within a region, so that they may distribute that information to the other backbone routers. For a region that is also using ST or LS multicast routing internally, the backbone routers attached to the region will automatically receive the necessary information, as part of their participation in the internal multicast routing algorithm. For a region using one of our other algorithms internally, additional mechanisms must be employed to detect group appearances and disappearances, such as a variant of the Host Membership Protocol that operates between internal routers and backbone routers. Note that this requirement imposed by inter-region ST or LS multicast routing affects all lower levels of the hierarchy, not just the immediately subordinate level.

The DV multicast algorithm, when used for inter-region routing, allows more flexibility in the choice of intra-region algorithms. It acquires its knowledge of group memberships (actually, the *absence* of group memberships) on demand, by sending multicast packets to all regions and getting prune messages in response. That means that the backbone routers need not learn about group memberships within a region before there is any traffic for those groups, and that the DV multicast algorithm may be used within the region, as an alternative to ST or LS multicast, without needing any additional membership discovery mechanisms.

## 9.2 Scoped Groups

As discussed in Section 3.2, the Host Group Model includes the notion of *scoped groups*, that is, groups for which all members and senders are limited to a particular administrative domain. For example, some groups might have a scope limited to a single site or campus. From a network management perspective, it is most convenient if the administrative domain boundaries correspond to regional boundaries in the routing hierarchy. That is usually the case for the "site" domain, and for other potentially useful domains such as "nation" or "continent". However, there may well be routing boundaries that do not define useful administrative domains, and there may be administrative domains that do not map directly onto routing boundaries.

The enforcement of scope limits occurs during multicast packet forwarding. A router that is on an administrative scope boundary must be configured to know about that boundary and must take that boundary into account when forwarding multicast packets, based on the scope of the packets' destination addresses.

When the scope boundaries do correspond to hierarchical routing boundaries, the routers can take advantage of scoping to avoid unnecessary costs. In particular, there is no need for the routers within a region to forward copies of multicast packets to any backbone routers if the scope of those packets is known not to extend beyond the region. Also, backbone routers need not, and should not, distribute membership information to other backbone routers concerning groups whose scope is limited to one region.

## 9.3 Chapter Summary

Our multicast routing algorithms may be used hierarchically, to eliminate the dominant scaling limit of the per-subnetwork costs. Two of our algorithms—the DV and LS multicast algorithms—have additional caching costs that grow with the number of active multicast (source, group) pairs. Hierarchical routing enables all the sources in the same region to be treated as a single source, yielding the same scaling benefit for that term of the caching costs. For the non-hierarchical use of those algorithms, we argued that the number of groups to which any one source would be actively sending would not significantly increase as the internetwork grew, so that the caching costs would be almost directly proportional to the number of sources. This is less likely to be true for the use of the algorithms at the top levels of a hierarchy—as the internetwork grows by adding more sub-regions and sub-sub-regions, the number of groups for which each region will originate packets is expected to grow. Much of this effect, however, can be alleviated by the use of scoped groups: any growth in the number of non-global destinations incurs no cost in the top-level routers.

Of course, as an internetwork grows, the amount of multicast traffic destined to global groups can be expected to grow significantly, for example, from applications such as videoconferencing. However, the bandwidth cost of conveying that traffic should not be considered as overhead, but rather should be compared to the much greater bandwidth that would be required to do the same thing with unicast alone. We expect that the bandwidth saved by using multicast delivery will compensate many times over for the management costs of supporting global groups.

# Chapter 10

# Other Multicast Routing Issues

Here, we discuss some secondary issues that pertain to internetwork multicast routing, independent of the specific routing algorithms described in the previous chapters.

## 10.1 Multihomed Hosts

A *multihomed host* is a host attached to more than one subnetwork. When a multihomed host sends a multicast packet, the question arises: should it transmit a copy of the packet on each of its attached subnetworks, or only on one? The answer depends on the nature of the information maintained by the routers, as follows:

- In most internetworks, such as those based on DARPA IP gateways or IEEE 802 datalink bridges, the routers do not maintain enough topological information to know that a host is multihomed—a multihomed host looks, to the routers, like multiple, separate hosts, each with its own (unicast) address on a different subnetwork. If a multihomed host on such an internetwork sends a copy of a multicast packet to each of its attached subnetworks, the routers have no way of recognizing that the separate copies come from the same host, and they have no choice but to treat them as multiple, independent multicasts.

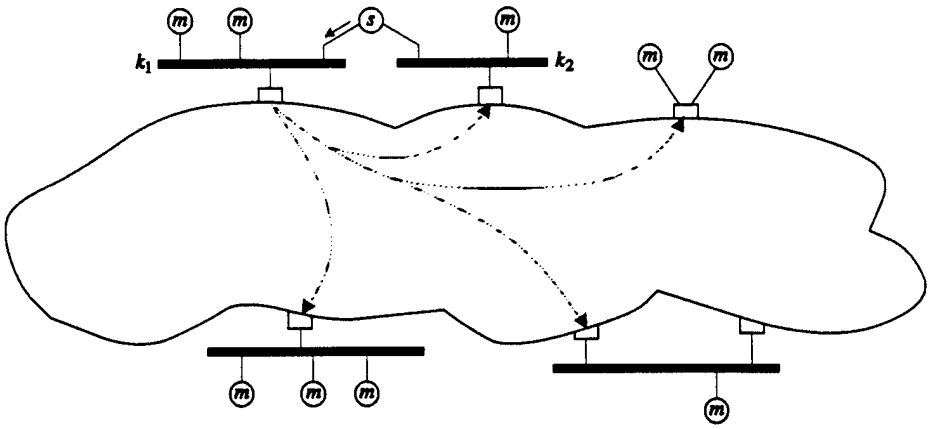
  Figure 10.1(a) shows how, in the case where the routers are unaware of multihoming, a multicast packet sent from a multihomed host, $s$, to one of its attached

subnetworks, $k_1$, is delivered to all other subnetworks containing members, $m$, of the destination group, including $s$'s other attached subnetwork, $k_2$. If $s$ were to transmit a copy of the packet on $k_2$ as well, it would be delivered as a duplicate to all group members, including those on $k_1$. Since duplicate delivery is often undesirable, the best way for a multihomed host in such an internetwork to handle a multicast *Send* request by an upper-layer protocol is to transmit the multicast packet on only one attached subnetwork. The upper-layer protocols may choose which subnetwork is used, and may invoke the *Send* operation multiple times with the same packet data but a different choice of subnetwork, if that should be desired in a specific case. (An example where it might be desired is when performing an expanding-ring search, where the multicast packets are likely to exceed their hop limits before they can be delivered as duplicates).

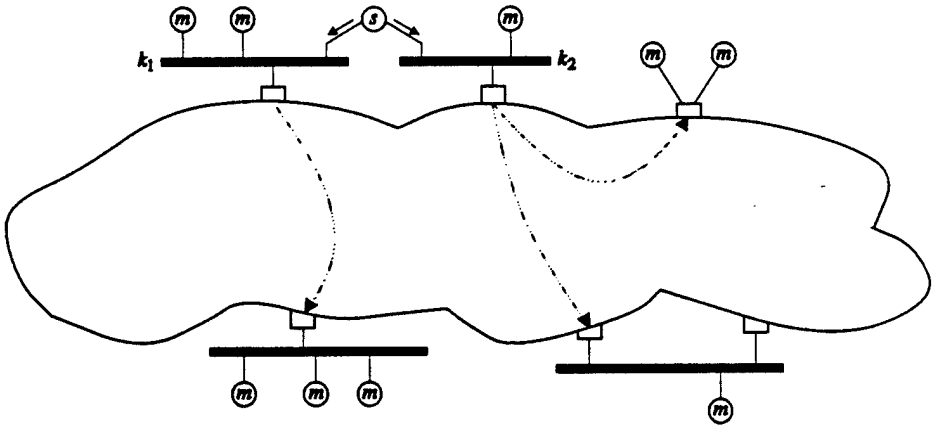
- In some internetworks, the routers *do* maintain enough topological information to know that a host is multihomed. That is the case, for example, for "Level 1 Intermediate Systems" in the IS-IS routing protocol [45] proposed for ISO CLNP. Under that protocol, a host attached to more than one subnetwork in the same Level 1 area may be assigned a single address (NSAP) in that Area, and the routers learn of the host's presence on each subnetwork through operation of the ES-IS protocol [44]. Since the routers can tell when the origin of a multicast packet is a multihomed host (and *cannot* tell, from its source address, the starting subnetwork of the multicast packet), they can (and must) assume that the host has taken care of delivery on each of its attached subnetworks, and they need only complete delivery to the remaining member subnetworks, as illustrated in Figure 10.1(b).

  Having the source host transmit multicast packets to all attached subnetworks is attractive, for the following reasons:

  - It relieves the upper-layer protocols of any concern about multiple subnetwork attachments when sending a multicast packet—the internetwork layer assumes all responsibility for generating multiple packet copies when the host is multihomed.




(a) routers unaware of multihoming




(b) routers aware of multihoming

Figure 10.1: A Multihomed Sender of a Multicast Packet

- When there are members of a destination group on more than one attached subnetwork, they all receive a multicast packet with a minimum of delay—none of them have to wait to receive a copy via a router.
- It better fits the desired behavior of hop-limited multicasting—a multicast packet sent with a hop-limit of one, for example, ought to be delivered to all members within one hop of the sender, regardless of what subnetwork those members are on.

However, the cost of supporting such behavior in the routers is larger routing tables and more routing packet overhead (on the order of the number of hosts rather than the number of subnetworks) which more severely limits the scalability of the (single-level) internetwork. Furthermore, it is not feasible to support such behavior for hosts that are attached to multiple subnetworks in *different* (single-level) internetworks; for example, a CLNP host connected to two different Level 1 IS-IS areas or to two entirely different routing domains must have a different identity (NSAP) in each area or domain, for use when originating or receiving packets via those areas or domains.

The distance-vector and link-state multicast routing algorithms we have described work properly in either of these two cases, depending only on the contents of the unicast routing data structures in which they look up the sources of multicast packets. Our spanning-tree algorithms, however, are not able to take advantage of multihoming knowledge, since multihoming violates the spanning-tree assumption that there is only one path to any single host from any other host.

Multihoming also introduces some slight complication on the receiving end of a multicast packet, that is, when a member of a destination host group is attached to more than subnetwork. As specified in the Host Group Model service interface (Section 3.5), when a multihomed host joins a host group, it may choose to join the group on one or more of its attached subnetworks. When it joins on more than one subnetwork, it will receive duplicates of any multicast packets that are sent to its group with a sufficiently large hop limit to reach each of those subnetworks, as shown in Figure 10.2. However, there are cases where this may be tolerable or even

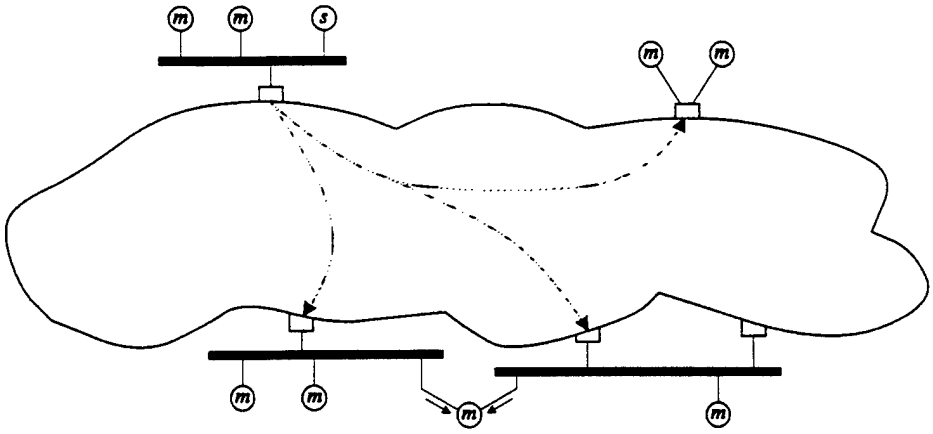



Figure 10.2: A Multihomed Member of a Host Group

desirable. For example, a host supporting a critical application might be multihomed explicitly for the purpose of increasing the probability of successful reception of multicast packets, and therefore would be willing to incur the cost of duplicate reception. Another example is a service that is to be located by expanding-ring search—if it is attached to more than one subnetwork, it is preferable for it to be reachable within one hop from hosts on any of those subnetworks. The service interface therefore allows the application to choose how many, and which, subnetworks to include in a group membership.

## 10.2 Multipath Routing

For all of our multicast routing algorithms, as long as the topology is not changing, there is a single delivery path from a source of a multicast packet to each member of its destination group; the set of such paths form a single delivery tree for the particular (source, group) pair. However, it can be beneficial in some circumstances to allow different packets from a given source to a given destination member to follow different paths. In particular:

- Different packets may demand different "qualities of service" (QOS) that can be satisfied only by sending them along different paths. For example, both DARPA IP and ISO CLNP include a field in the internetwork packet header

for specifying which particular measure (e.g., delay, throughput, reliability, expense) the sender wishes optimized when delivering the packet. For unicast routing, this capability is provided by (logically) running multiple instances of the unicast routing algorithm in parallel, each using a different set of weights (delay, bandwidth, etc.) for the subnetworks. The output of the algorithms is a set of routing tables in each router; the QOS field in each packet is used to select an appropriate routing table for that packet. Since our multicast routing algorithms are built upon the unicast routing data structures, it is straightforward to provide separate delivery trees for different (source, group, QOS) triples, simply by using the appropriate underlying data structures for each different QOS. (Note that, for our distance-vector and link-state multicast algorithms which cache information about active multicast trees, supporting multiple QOS values does *not* imply a corresponding multiplication of cache size, since all multicast packets from a particular source to a particular group are likely to be sent with the same QOS.)

- For unicast routing algorithms that deliver packets along shortest paths (e.g., the distance-vector or link-state algorithms, but not the spanning-tree algorithm), there may be more than one shortest path from a given source to a given destination (for a given QOS). Most implementations of those algorithms simply choose one of the paths, and send all packets with the given source and destination along that one path; some implementations exploit the multiple paths, sending subsequent packets along different paths in order to increase throughput or reduce congestion. In the case of multicast routing, there may be more than one shortest-path *tree* from a given source to a given destination group. However, the algorithms we have described for distance-vector and link-state multicasting rely on choosing a single tree for any particular (source, group) pair; all routers must choose the *same* tree, in order to avoid packet loops and to accomplish delivery to all group members. (For distance-vector routing, this is ensured by the tie-breaking algorithm used when identifying "child" subnetworks. For link-state routing, it is ensured by having all routers perform the same Dijkstra SPF algorithm on identical data structures.) Hence,

our algorithms are not able to exploit multiple shortest-path trees for multicast delivery. We do not consider this to be a serious drawback, for the following reasons:

- For many applications of multicast, the rate of multicast transmission is very low (for example, when using multicast to locate or advertise services), and therefore not a significant contributor to the bandwidth and congestion problems that multipath routing alleviates.
- For high-rate multicast applications such as real-time voice or video conferencing, it is usually preferable to constrain all packets to follow a single path, to any one destination, in order to minimize the variance of delivery delay (i.e., "jitter").
- Although transport layer protocols for datagram internetworks can handle the packet re-ordering that normally occurs when doing multipath routing, the protocol implementations are usually optimized for the case of in-order delivery and suffer degraded performance when a significant percentage of packets arrive out of order. For this reason, we expect to see *less* use of multipath routing for unicast packets, especially as long-distance bandwidth becomes more plentiful.

Also, it should be noted that multipath routing performed at the *subnetwork* layer, for example, by grouping multiple "trunk" circuits into one logical link, works fine with our multicast routing algorithms.

## 10.3 Multicast Reliability and Performance

In the Host Group Model, we specified that multicast packets be delivered with reliability and performance *close* to that of unicast packets sent to the same destinations. We have satisfied that requirement in our various routing algorithms simply by ensuring that multicasts are forwarded along the same (or equally as good) paths as unicasts. However, even when the paths are the same, the reliability and performance of multicast delivery may not reach that of unicast delivery, for the following reasons:

- During periods when paths are being changed immediately following a topology change, multicast packets that happen to be in flight have a lower probability of reaching their destinations than do unicast packets. That is because the forwarding decision for a unicast packet depends only on where the packet is going (i.e., its destination address)—if it gets sent in the wrong direction by out-of-date routers, it might still reach its destination via other routers whose tables are up-to-date or by looping until the out-of-date routers eventually learn of the new topology. The forwarding decision for a multicast packet, on the other hand, depends on where the packet came from (i.e., its source address and which interface it arrived on) as well as where it is going (its destination group)—a copy of a multicast packet sent in the wrong direction by an out-of-date router is most likely to arrive at another router via the wrong interface, in which case it will be dropped. In most operational internetworks small enough to be treated as a single-level, flat routing domain, topology changes should be sufficiently rare, and periods of routing inconsistency sufficiently brief, that this difference between unicast and multicast reliability will be insignificant.

- The delivery delay and throughput for multicasts to a particular member might be slightly greater than that of unicasts to the same member, due to the extra processing time required by the routers when they must generate multiple packet copies at branch points in the multicast delivery tree. This extra delay can be made negligible by appropriate implementation techniques in the routers, such as allowing a single, shared packet buffer to be queued for concurrent transmission on more than one interface.

# Chapter 11

# Conclusions

We conclude this dissertation with a discussion of the relative merits of our various multicast routing protocols and some criteria for choosing among them, a short list of outstanding problems in the area of internetwork multicasting, and a summary of our main contributions.

## 11.1 Comparison of the Algorithms

In Chapters 6, 7, and 8, we presented six new algorithms for routing multicast datagrams in a single-level internetwork:

- Spanning-Tree (ST) Truncated-Broadcast
- Spanning-Tree (ST) Multicast
- Distance-Vector (DV) Truncated-Broadcast
- Distance-Vector (DV) Multicast
- Link-State (LS) Truncated-Broadcast
- Link-State (LS) Multicast

Each of the algorithms satisfies the service requirements of the Host Group Model defined in Chapter 3, in terms of:

**Functionality:** Each algorithm supports multicasting from any host to any host group, where the members of the group may be distributed across any number of subnetworks, and may join and and leave the group independently, at any time.

**Reliability:** Each algorithm supports delivery of a multicast packet to each of its destination group members with reliability close to that of a unicast packet sent to the same member. This is accomplished by delivering a multicast packet to a host via the same (or an equally good) path as used for delivering a unicast packet from the same source; as long as that path does not change while the packet is "in flight", both multicasts and unicasts are subjected to the same probability of loss, damage, or duplication.

**Performance:** Each algorithm supports delivery of a multicast packet to each of its destination group members with delay and throughput close to that of a unicast packet sent to that same member, as a consequence of routing the multicast via the same (or an equally good) path as a unicast to the same member.

The overhead costs associated with each of the algorithms are modest, relative to the resources available in typical current (non-hierarchical) internetworks, and those costs scale as well as the underlying unicast routing costs, or better. The algorithms may be used hierarchically to extend multicast service across very large internetworks, with different algorithms being used in different regions and at different levels of the hierarchy.

We expect that, in most internetworks, concerns other than multicast will determine which underlying routing algorithm (spanning-tree, distance-vector, or link-state) is used; that is why we have developed multicast routing schemes compatible with each of the basic routing algorithms. After that determination, there is still a choice to be made between truncated-broadcast and multicast routing. Truncated-broadcast routing is best suited to topologies that have significantly more bandwidth available in their interior subnetworks than in their leaf subnetworks (such as an internetwork composed of one or more FDDI backbones interconnecting a collection of leaf Ethernets), or for routing to the following types of host groups:

- Groups that have members on all or almost all subnetworks.
- Groups to whom multicast packets are rarely sent.
- Groups to whom multicast packets are sent only with very small hop limits.

Multicast routing is the better choice for internetworks of more general topology, and for routing to groups that do not satisfy any of the above listed properties.

(It might even be desirable to use both truncated-broadcast and multicast routing in the same internetwork, using truncated-broadcast to reach groups of the types listed above, and multicast to reach any other groups. That would be straightforward to do if the two categories of groups could easily be distinguished by their group addresses. However, the additional constraints that would impose on group address allocation, and the problems that might arise when a group changes from one category to the other, suggest that such an approach be tried only after measurements of real multicast traffic in real internetworks show that it would be worthwhile.)

In single-level internetworks, and at the lowest levels of hierarchical internetworks, if the choice of underlying routing algorithm is not pre-determined, we recommend the use of the link-state multicast algorithm for the following reasons:

- The link-state multicast algorithm is able to provide shortest-path routing, even in the presence of asymmetric path weights, unlike the distance-vector and spanning-tree algorithms.
- The link-state multicast algorithm is able to pre-emptively discard multicast packets that, due to hop limits, would expire before reaching some destination members, thus saving some bandwidth. (That is likely to be a common occurrence when multicast is being used for expanding-ring searching.) Under the other algorithms, such packets are forwarded until their hop counts expire.
- Under the link-state algorithm, the costs for supporting the activity of a given (source, group) pair are borne by those routers that are on the delivery tree for multicasts from that source to that group, as would be expected. Under the distance-vector algorithm, on the other hand, costs are incurred by all of

the routers that are *not* on the delivery tree, and no costs are borne by the routers internal to the delivery tree, which is counterintuitive. This issue may be important in environments in which internetwork resource usage must be charged to the users, where it is important that the charges match the users' expectations.

Independent of multicast considerations, link-state routing has several other advantages over distance-vector routing, and many internetworks have been converted from distance-vector to link-state routing to enjoy those advantages [50].

At higher levels in a hierarchical internetwork, the advantages of link-state multicast routing listed above may be considered less significant, and distance-vector routing may be preferred because of the greater flexibility it offers for choice of lower-level multicast algorithms, as discussed in Section 9.1.

## 11.2 Outstanding Problems

The availability of a widespread internetwork multicast service raises a number of important concerns that have not been addressed in this dissertation. Here are three:

- *Congestion control* for multicast traffic is more complicated than for unicast traffic. Many of the techniques developed for unicast congestion control involve feedback signals to the sender, which do not work well (or can actually increase congestion) for senders to large multicast groups.

- *Privacy* of data traffic is a more serious concern for some multicast traffic than for unicast traffic, because of the ease with which hosts can join groups, and the lesser control over, and knowledge about, where multicast packets are sent. However, most of the work to date in encryption protocols applies only to communication between two peers.

- *Accounting* for multicast communication costs is considerably more complicated than for unicast, and it seems likely that more than one charging model will be required, based on such factors as whether the senders or the "subscribers"

should pay for particular multicast transmissions, and whether or not there should be a charge just for belonging to a group, independent of packets received.

It should be noted that these issues are still far from being resolved even for *unicast* service, in the types of datagram internetworks for which our protocols have been designed.

## 11.3 Summary of Main Contributions

The main contributions of this dissertation are the following:

- A new multicast service model for datagram internetworks, that (a) supports a wider range of multicast applications than previously implemented or proposed models, (b) is a natural extension of the unicast service model offered by datagram internetworks, and (c) is amenable to efficient implementation in large-scale internetworks. Two novel features of the service model are the notion of scoped groups, important for the scalability of the service, and the use of hop limits for expanding-ring searches.

- A set of new multicast routing algorithms, based on existing unicast algorithms, that have low overhead, high performance, and scalability as good as, or better than, unicast routing.

- A scheme for combining multicast routing algorithms hierarchically, in order to provide multicast service across very large-scale internetworks.

# Bibliography


[1] A. Aggarwal. Protocol standard for a NetBIOS service on a TCP/UDP transport: Detailed specifications. RFC 1002, SRI Network Information Center, March 1987.

[2] L. Aguilar. Datagram routing for internet multicasting. In *Proc. ACM SIGCOMM '84 Communications Architectures and Protocols*, pages 58–63. ACM, June 1984.

[3] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *Data Structures and Algorithms.* Addison-Wesley, Reading, Mass., 1983. Dijkstra's algorithm.

[4] F. Backes. Transparent bridges for interconnection of IEEE 802 LANs. *IEEE Network*, 2(1):5–9, January 1988.

[5] N. E. Belkeir and M. Ahamad. Low cost algorithms for message delivery in dynamic multicast groups. In *Proc. 9th Intl. Conf. Distributed Computing Systems*, pages 110–117. IEEE, June 1989.

[6] Bell Communications Research, Inc. Generic system requirements in support of switched multi-megabit data service. Technical Advisory TA-TSY-000772, Issue 3, October 1989.

[7] R. E. Bellman. *Dynamic Programming.* Princeton University Press, Princeton, N.J., 1957.

[8] K. Bharath-Kumar and J. M. Jaffe. Routing to multiple destinations in computer networks. *IEEE Transactions on Communications*, COM-31(3):343–351, March 1983.

[9] K. P. Birman, R. Cooper, and B. Gleeson. Programming with process groups: Group and multicast semantics. Technical Report 91–1185, Department of Computer Science, Cornell University, January 1991.

[10] K. P. Birman and T. A. Joseph. On communication support for fault tolerant process groups. RFC 992, SRI Network Information Center, November 1986.

[11] D. R. Boggs. *Internet Broadcasting.* PhD thesis, Electrical Engineering Dept., Stanford University, January 1982. Also Tech. Rep. CSL-83-3, Xerox PARC, Palo Alto, Calif.

[12] D. R. Boggs, J. F. Shoch, E. A. Taft, and R. M. Metcalfe. PUP: An internetwork architecture. *IEEE Transactions on Communications*, COM-28(4):612–624, April 1980.

[13] R. T. Braden and J. B. Postel. Requirements for Internet gateways. RFC 1009, SRI Network Information Center, June 1987.

[14] J. L. Carrol, D. D. E. Long, and J-F. Paris. Block-level consistency of replicated files. In *Proc. 7th International Conference on Distributed Computing Systems*, pages 146–153. IEEE, September 1987.

[15] J. D. Case, M. Fedor, M. L. Schoffstall, and C. Davin. Simple network management protocol (SNMP). RFC 1157, SRI Network Information Center, May 1990.

[16] J-M. Chang. Simplifying distributed database systems design by using a broadcast network. In *Proc. ACM SIGMOD '84*, pages 223–233. ACM, June 1984.

[17] D. R. Cheriton. VMTP: A transport protocol for the next generation of communications systems. In *Proc. ACM SIGCOMM '86 Symposium on Communications Architectures and Protocols*, pages 406–415. ACM, August 1986.

[18] D. R. Cheriton. VMTP: Versatile message transaction protocol—protocol specification. RFC 1045, SRI Network Information Center, February 1988.

[19] D. R. Cheriton and M. Stumm. Multi-satellite star: Structuring parallel computations for a workstation cluster. *Distributed Computing*, 1987. To appear.

[20] D. R. Cheriton and W. Zwaenepoel. Distributed process groups in the V kernel. *ACM Transactions on Computer Systems*, 3(2):77–107, May 1985.

[21] I. Chlamtac and S. Kutten. On broadcasting in radio networks — problem analysis and protocol design. *IEEE Transactions on Communications*, December 1985.

[22] D. Clark. The design philosophy of the DARPA internet protocols. In *Proc. ACM SIGCOMM '88 Symposium.* ACM, August 1988.

[23] W. Croft and J. Gilmore. Bootstrap protocol (BOOTP). RFC 951, SRI Network Information Center, September 1985.

[24] Y. K. Dalal and R. M. Metcalfe. Reverse path forwarding of broadcast packets. *Communications of the ACM*, 21(12):1040–1048, December 1978.

[25] S. E. Deering. Host extensions for IP multicasting. RFC 1112, SRI Network Information Center, August 1988.

[26] S. E. Deering. Multicast routing in internetworks and extended LANs. In *Proc. ACM SIGCOMM '88 Symposium.* ACM, August 1988.

[27] Digital Equipment Corporation. *DECnet Digital Network Architecture (Phase IV) General Description.* document no. AA-149A-TC.

[28] Digital Equipment Corporation, Intel Corporation, and Xerox Corporation. The Ethernet: A local area network; data link layer and physical layer specifications, version 1.0. *Computer Communications Review*, 11(3):20–66, September 1980.

[29] Digital Equipment Corporation, Northern Telecom, Inc., and StrataCom, Inc. Frame relay specification with extensions, based on proposed t1s1 standards. Document Number 001-208966, Revision 1.0, September 1990.

[30] G. Falk et al. Integration of voice and data in the wideband packet satellite network. *IEEE Journal on Selected Areas in Communication*, December 1983.

[31] L. R. Ford Jr. and D. R. Fulkerson. *Flows in Networks.* Princeton University Press, Princeton, N.J., 1962.

[32] J. W. Forgie. ST—a proposed internet stream protocol. IEN 119, SRI Network Information Center, September 1979.

[33] M. C. Hamner and G. R. Samsen. Source routing bridge implementation. *IEEE Network*, 2(1):33–36, January 1988.

[34] J. Hart. Extending the IEEE 802.1 MAC bridge standard to remote bridges. *IEEE Network*, 2(1):10–25, January 1988.

[35] W. R. Hawe, M. F. Kempf, and A. J. Kirby. The extended local area network architecture and LANBridge 100. *Digital Technical Journal*, (3):54–72, September 1986.

[36] C. Hedrick. Routing information protocol. RFC 1058, SRI Network Information Center, June 1988.

[37] IEEE. Standards for local area networks: Logical link control. ANSI/IEEE Standard 802.2-1985 (ISO/DIS 8802/2), 1985.

[38] IEEE. Standards for local area networks: Token ring access method and physical layer specifications. ANSI/IEEE Standard 802.5-1985 (ISO/DIS 8802/5), 1985.

[39] IEEE. Draft standard p802.1d/d6, MAC bridges, September 1988.

[40] International Business Machines Corp. *Technical Reference PC Network.* document 6322916.

[41] International Organization for Standardization (ISO). *International Standard 7498, Information Processing Systems—Open Systems Interconnection—Basic Reference Model*, October 1984.

[42] International Telegraph and Telephone Consultative Committee (CCITT). Draft recommendation I.211—B-ISDN service spects. Study Group XVIII—Report R 34, June 1990.

[43] ISO/IEC JTC1/SC 21, Secretariat USA (ANSI). *ISO 7498-1/PDAD2—Proposed Draft Addendum 2 to ISO 7498-1 on Multipeer Data Transmission*, January 1988.

[44] ISO/IEC JTC1/SC 6, Secretariat USA (ANSI). *ISO 9542—End System to Intermediate System Routeing Exchange Protocol for Use in Conjunction with ISO 8473*, March 1988.

[45] ISO/IEC JTC1/SC 6, Secretariat USA (ANSI). *Intermediate System to Intermediate System Intra-Domain Routeing Exchange Protocol for Use in Conjunction with ISO 8473*, October 1989.

[46] ISO/TC 97/SC 6, Secretariat USA (ANSI). *ISO 8473—Protocol for Providing the Connectionless-Mode Network Service*, May 1987.

[47] L. Kleinrock and F. Kamoun. Hierarchical routing for large networks; performance evaluation and optimization. *Computer Networks*, 1:155–174, 1977.

[48] T. P. Mann. *A Decentralized Naming Facility.* PhD thesis, Computer Science Dept., Stanford University, 1987.

[49] P. K. McKinley and J. W. S. Liu. Multicast tree construction in bus-based networks. *Communications of the ACM*, 33(1):29–42, January 1990.

[50] J. M. McQuillan, I. Richer, and E. C. Rosen. The new routing algorithm for the ARPANET. *IEEE Transactions on Communications*, COM-28(5):711–719, May 1980.

[51] J. M. McQuillan and D. C. Walden. The ARPANET design decisions. *Computer Networks*, 1, August 1977.

[52] P. V. Mockapetris. Domain names—implementation and specification. RFC 1034, SRI Network Information Center, November 1987.

[53] J. Mogul. Broadcasting internet datagrams. RFC 919, SRI Network Information Center, October 1984.

[54] J. Moy. The OSPF specification. RFC 1131, SRI Network Information Center, October 1989.

[55] R. Perlman. An algorithm for distributed computation of a spanning tree in an extended LAN. In *Proc. 9th Data Communications Symposium*, pages 44–53. ACM/IEEE, September 1985.

[56] J. Postel. Internetwork protocol approaches. *IEEE Transactions on Communications*, April 1980.

[57] J. Postel. Internet protocol. RFC 791, SRI Network Information Center, September 1981.

[58] Protocol Engines, Inc. XTP protocol definition. Revision 3.4, 1989.

[59] S. K. Sarin. Interactive on-line conferences. Technical Report MIT/LCS/TR-330, MIT Laboratory for Computer Science, December 1984.

[60] M. Satyanarayanan and E. H. Siegal. MultiRPC: A parallel remote procedure call mechanism. Technical Report CMU-CS-86-139, Carnegie-Mellon University, August 1986.

[61] G. S. Sidhu, R. F. Andrews, and A. B. Oppenheimer. *Inside AppleTalk.* Apple Computer Inc., July 1986.

[62] W. D. Sincoskie and C. J. Cotton. Extended bridge algorithms for large networks. *IEEE Network*, 2(1):16–24, January 1988.

[63] Sun Microsystems. *Remote Procedure Call Reference Manual.* Mountain View, California, October 1984.

[64] C. Topolcic. Experimental internet stream protocol, version 2 (ST-II). RFC 1190, SRI Network Information Center, October 1990.

[65] D. W. Wall. *Mechanisms for Broadcast and Selective Broadcast.* PhD thesis, Electrical Engineering Dept., Stanford University, June 1980. Also Tech. Rep. 190, Computer Systems Lab., Stanford.

[66] A. G. Waters, C. J. Adams, I. M. Leslie, and R. M. Needham. The use of broadcast techniques on the Universe network. In *Proc. ACM SIGCOMM '84 Communications Architectures and Protocols*, pages 52–57. ACM, June 1984.

[67] Xerox Corporation. Internet transport protocols. XSIS 028112, Xerox, Stamford, Connecticut, December 1981.